12-2013

Tracking Real-Time Nanoparticle Positions and Measuring Three-Dimensional Solution Flow with a Four-Focus Confocal Microscope

James Andrew Germann
University of Tennessee - Knoxville, jgermann@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by James Andrew Germann entitled "Tracking Real-Time Nanoparticle Positions and Measuring Three-Dimensional Solution Flow with a Four-Focus Confocal Microscope." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Physics.

Lloyd M. Davis, Major Professor

We have read this dissertation and recommend its acceptance:

Christian G. Parigger, William H. Hofmeister, Marianne Breinig

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Tracking Real-Time Nanoparticle Positions and Measuring Three-Dimensional Solution Flow with a Four-Focus Confocal Microscope

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

James Andrew Germann
December 2013

ACKNOWLEDGEMENTS

I would like to first thank my committee. Without their valuable insight and guidance, none of this would have been possible. I would especially like to thank my advisor Dr. Lloyd Davis. His knowledge, experience, and most importantly patience provided the best guidance I could have hoped for. I would also like to thank Dr. Brian Canfield and Jason King. Their help both in and out of the lab was worth its weight in gold.

To my friends, thank you for being around for the good times and the not so good times. Thanks especially to William Robinson, Matthew Duran, and James Thomas. You were always there with a kind word and an impetus when I really needed it.

Last but not least, thank you to my family. You've all been an inspiration.

ABSTRACT

This dissertation presents the development of instrumentation for measuring the position of a single emitter within the sample volume of a confocal fluorescence microscope with sub-diffraction limited precision in three dimensions together with applications for determining solution flow and for tracking a fluorescent nanoparticle as it undergoes Brownian diffusion. The localization method is based on comparing photon counts from alternating excitation of the emitter by four laser beams, which are focused at slightly offset positions in a tetrahedral pattern within the confocal volume. Two experimental set-ups are constructed. In the first, the four beams are from a femtosecond laser, which provides two-photon excitation (2PE) of a nanomolar solution of rhodamine B. Time-resolved photon counting into four channels and cross-correlation of the channels yields a set of sixteen curves that vary with the flow of solution through the tetrahedral pattern. Standard fluorescence correlation spectroscopy fitting methods are extended to model solution flow and a Gaussian-Lorentzian beam profile for 2PE. Global fitting of the sixteen curves to extract the flow is studied using data from a collaborator's computer simulation. The model successfully fits simulated data with flow in one dimension, but accuracy is found to be poor when simultaneously fitting three velocity components. In the second set-up, the four beams are provided by 635-nm fiber-coupled laser diodes. LabVIEW real-time software alternately pulses the diodes, performs time-gated photon counting to estimate the nanoparticle position, and controls a piezo stage to track its motion as it undergoes diffusion. The confocal microscope enables tracking of nanoparticles with diffusivity exceeding 10 μm^2 [micron^2] s^{-1}, currently limited by the 1.8 ms update interval of the piezo stage, while also providing exceptional sensitivity, with a

net fluorescence detection efficiency at least 4.2%, comparable to that of a highly optimized single-molecule microscope.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I-INTRODUCTION

Through the use of a series of optical components, four laser beam foci with adjustable positions can be created within the probe volume of a confocal fluorescence microscope. Fluorescence generated by the sample within the probe volume can be separated by focus into four channels by alternately pulsing the laser foci and using time-gated or time-resolved photon counting. With a minimum of four suitably placed laser foci, a three-dimensional space is defined. In this research, the four-focus setup is examined in two capacities—for measuring flow velocity components in three-dimensions, with the use of two-photon excitation, and for measuring the position of a single emitter to track its motion as it undergoes Brownian diffusion in solution, with the use of one-photon excitation.

The dissertation begins by examining the application of the four-focus microscope with fluorescence correlation spectroscopy (FCS) to the measurement of flow. A short background of previous work will be given, followed by a description of simulations modeling four-focus measurements of flow. The results of the simulations are then discussed. Background on single-molecule localization is then presented. The apparatus for the four-focus localization experiment will be described and results of tracking fluorescent latex beads will follow. Fluorescence throughput measurements show a collection efficiency at least 4.2%, denoting the four-focus microscope as single molecule sensitive. At the end, some concluding remarks will be offered.

CHAPTER II-BACKGROUND

2.1 Fluorescence Correlation Spectroscopy

The first FCS experiments were reported in 1972 by Magde, Elson, and Webb as a way of probing the mechanisms occurring in a thermal equilibrium state without perturbing the sample, [1-3] specifically the kinetic reaction rate of ethidium bromide to DNA in those first experiments. The kinetic rate was measured from the fluctuations away from equilibrium, as determined from the fluctuations in the fluorescence. Besides chemical kinematic rates, FCS is used to gauge the rate of diffusion of molecules in solution by measuring the autocorrelation function of the fluorescence fluctuations as molecules diffuse in and out of the laser focus.[2] These initial experiments were an effective measure of motion in two dimensions (2D), such as diffusion of molecules on a membrane, and the addition of a pinhole for confocal detection added depth resolution.[4]

FCS experiments can also measure the flow velocity of a solution. The autocorrelation function can determine the magnitude of flow velocity,[5] but not the direction. The addition of a second focus displaced from the first gives the component of flow velocity along the direction of displacement.[6] FCS experiments have also implemented two-photon excitation, which results in a self-selecting confocal probe volume with depth resolution without the use of a pinhole.[7] Two-photon excitation FCS has been used to measure velocity with one-focus[8] and two foci.[9] The research presented in this half of the dissertation focuses on a FCS experiment that measures the three flow velocity components simultaneously using four spatially separated laser foci and two-photon excitation. The functional form of the correlation functions for fitting experimental FCS data obtained with flow through multiple laser foci is derived for a

Gaussian-Lorentzian[7, 9, 10] model of the laser beam profile at each focus. Initial experimental data is presented with results from a simulation study, which compares the Gaussian-Lorentzian and three-dimensional (3D) Gaussian model.

2.2 One-Photon and Two-Photon Confocal Microscopy

2.2.1 One-Photon Confocal Microscopy

In one-photon microscopy, a photon is absorbed by a fluorescent particle and the particle becomes excited. After a short period of time (typically on the order of nanoseconds) a photon is emitted by the particle.[11] The energy of the emitted photon is slightly less than the energy of the absorbed photon due to energy lost in vibrational states,[11] known as the Stokes shift. A pinhole is used to limit the amount of background light, scattered light, and fluorescence generated outside the focal plane,[4, 12, 13] increasing the signal-to-noise ratio of the microscope system.[14] One-photon confocal microscopy has been used to study multiphase characteristics of cell walls,[15] hindered diffusion in hydrogels,[16] and the binding rates of molecules to proteins.[17]

The advantage of one-photon microscopy is that the laser source exciting the particles can be a low-powered, continuous-wave laser. In conjunction with a sensitive single-photon avalanche diode (SPAD), fluorescence can be detected from a stable fluorescent species with only a few microwatts of power. However, the separation of fluorescence signal from excitation light can be difficult should the Stokes shift be small.

2.2.2 Two-photon Confocal Microscopy

In two-photon microscopy, two photons are absorbed within a very short period of time to excite a fluorescent particle (~15 fs).[18] The short time available for the second photon's absorption requires that a high density of photons arrive at the sample at approximately the same time. Pulsed lasers are typically used to provide this high photon density,[19] although a continuous-wave laser can induce a two-photon process if its intensity is two orders of magnitude greater than the pulsed beam.[20] The amount of fluorescence signal $F(t)$ generated by a two-photon process is

$$F\ t\ = \alpha \quad I^2\ r,t\ C\ r,t\ dV\ , \tag{2.1}$$

where α is the two-photon cross section, $I(r,t)$ is the intensity distribution of the focus, and $C(r,t)$ is the fluorescent species distribution in the sample.[7] As the signal depends on the square of the intensity, virtually only particles at the focus are excited, [7, 21-23] providing the depth selection previously stated.

This self-selectivity of the two-photon process eliminates the need for a pinhole,[22] although adding a pinhole can give a higher resolution.[23, 24] Restricting interactions to the focal plane also has the advantages of lessening photobleaching of fluorophores[21, 25] and limiting photodamage[21, 25] through the sample. Another advantage of two-photon excitation is that the optics used for fluorescence collection allow for collection of multiple emitted wavelengths without chromatic aberrations from the objective.[26] There is also the added benefit that the Stokes shift for two-photon excitation is generally, in terms of absolute value, greater than in one-photon excitation, making the fluorescence signal easier to separate from scattered and reflected excitation light.

Studies of biological systems have utilized two-photon microscopy because of its many advantages.[25, 27, 28] In addition to the reduced phototoxicity and natural depth selection of the two-photon process, near-infrared light typically has a greater penetration depth in biological samples than visible and ultraviolet light.[25, 27] With specialized collection optics, minimally invasive imaging of tissue inside living creatures at distances of millimeters below the surface has been demonstrated.[29] Two-photon microscopy has already found use in the study of calcium distributions in tissue,[25] mechanisms of the heart,[27] pH gradients of skin,[28] and the workings of the brain.[29]

2.3 Theory and Application of FCS

The correlation function is calculated from the fluorescence signal by comparing the times between photon arrivals, that is:

$$G_{ij}\ \tau\ =\ F_i\ t\ F_j(t+\tau)\ , \qquad 2.2$$

where $G_{ij}(\tau)$ is the correlation function, $F_i(t)$ is the signal recorded at time t, and τ is the delay between measurements.[7] The angled brackets denote the temporal average. The fluorescence signals can come from different sources, as denoted by the subscripts *i* and *j*. Examples of these different sources are separate observation volumes,[9] different detectors,[30] and varied fluorescence wavelength measurements.[31] When the sources of the fluorescence signals are different ($i \neq j$), the correlation functions are cross-correlations; when they are the same ($i = j$), the correlations are autocorrelations.

Equation 2.2 can be rewritten as

$$G_{ij}\ \tau\ =\ \delta F_i\ \tau\ \delta F_j(t+\tau)\ +\ F_i\ \ F_j\ , \qquad 2.3$$

where F is the average fluorescence signal and

$$\delta F = F\ t\ -\ F \tag{2.4}$$

is the instantaneous fluctuation of signal. Equation 2.3 can be normalized to the form

$$g_{ij} = \frac{\delta F_i(t)\delta F_j(t+\tau)}{F_i\ \ F_j} + 1. \tag{2.5}$$

Classically the fluctuation in signal corresponded to a fluctuation in current from a photomultiplier tube caused by the arrival of fluorescence photons,[2] but many modern techniques make use of SPADs and SPAD-like arrays to count individual photons as fluorescence signal.[31-34]

Although FCS can be used to determine rate constants and particle number, for now the theoretical formulation will concentrate on diffusion and flow through the excitation volume. As previously stated, the amount of fluorescence signal generated in a two-photon process is dependent on the square of intensity (eq. 2.1), while the fluorescence signal generated in a one-photon process is linearly dependent on the intensity:[27]

$$F\ t\ = \sigma\ \ \ I(r,t)C(r,t)dV \tag{2.6}$$

where σ is the one-photon cross section. The theoretical model of signal fluctuation (eq. 2.4) due strictly to changes in concentration from diffusional effects becomes

$$\delta F\ t\ = \sigma\ \ \ I\ r,t\ \delta C\ r,t\ dV \tag{2.7}$$

with the intensity assumed to be uniformly distributed over time.

2.3.1 3D Gaussian Model of Intensity and Correlation Functions

The model of intensity distribution can be approximated in a number of different ways. One model is the (3D) Gaussian model, where the transverse (in-plane) and longitudinal (optical axis) components are represented as Gaussians: [8]

$$I\ r,t\ = I_0 \exp\ \frac{-\ x-x_0}{2w_0^2}\ \exp\ \frac{-\ y-y_0\ ^2}{2w_0^2}\ \exp\ \frac{-\ z-z_0\ ^2}{2z_o^2}\ . \qquad 2.8$$

I_0 is the peak intensity, x_0, y_0, z_0 are the distances between the center of the laser beam focus and the origin along the *x*, *y*, and *z* axes, w_0 is the beam waist, and z_o is the distance along the optical axis where intensity falls off to e^{-2} of the peak. The two-photon correlation function is obtained by inserting eq. 2.8 into eq. 2.1 to get the two-photon fluorescence signal, which is then inserted into eq. 2.5. In addition, due to the ergodicity of a typical diffusing system,[35] the time average of fluorescence is equated to the spatial average of fluorescence as seen in eq. 2.9:

$$g_{ij} = 1 + \frac{\sigma^2}{F_i\ \ F_j} \qquad dr_i^3 dr_j^3 I_i^2\ r_i\ I_j^2\ r_j\ P\ r_i, r_j, \tau\ \ . \qquad 2.9$$

$P\ r_i, r_j, \tau\ $ is given by

$$P\ r_i, r_j, \tau\ \ =\ \ \delta C\ r_i, t\ \delta C\ r_j, t+\tau\ \ \ , \qquad 2.10$$

which describes the probability of a particle moving from position r_i to r_j, where $\delta C\ x,t\ $ is the instantaneous concentration fluctuation at point x for time *t*. The instantaneous concentration is determined from Fisk's Second Law:

$$\frac{\partial \delta C(r,t)}{\partial t} = D\nabla^2 \delta C\ r,t\ - v\delta C\ r,t\ \ , \qquad 2.11$$

where *D* is the diffusion coefficient and v is the flow velocity in solution. The solution to this differential equation for the initial condition of a particle at the origin at time $t = 0$ is

$$\delta C\ r,t\ dr = A \exp\ \frac{-(r-vt)^2}{4Dt}\ , \tag{2.12}$$

where *A* is a normalization constant. The probability from eq. 2.10 then becomes

$$\delta C(r_i,t)\delta C(r_j,t+\tau)\ =\ C\ (4\pi D\tau)^{-3\ 2} \exp\ \frac{-\ r_j-r_i\ -v\tau\ ^2}{4D\tau}\ , \tag{2.13}$$

and with the squared 3D Gaussian intensity

$$I^2\ r,t\ = I_0{}^2 \exp\ \frac{-\ x-x_0\ ^2}{w_0^2}\ \exp\ \frac{-\ y-y_0\ ^2}{w_0^2}\ \exp\ \frac{-\ z-z_0\ ^2}{z_o^2}\ , \tag{2.14}$$

the correlation function becomes

$$g_{ij} = 1 + \frac{C}{1+\frac{\tau}{\tau_D}\quad 1+\frac{w_o^2}{z_o^2}\ \frac{\tau}{\tau_D}} \exp\ \frac{-\ x_j-x_i-v_x\tau\ ^2}{w_o^2\ \ 1+\frac{\tau}{\tau_D}}$$

$$\times \exp\ \frac{-\ y_j-y_i-v_y\tau\ ^2}{w_o^2\ \ 1+\frac{\tau}{\tau_D}}\ \ \exp\ \frac{-\ z_j-z_i-v_z\tau\ ^2}{z_0^2\ \ 1+\ \frac{w_o^2}{z_o^2}\ \frac{\tau}{\tau_D}}\ , \tag{2.15}$$

where

$$\tau_D = \frac{w_o^2}{8D}. \tag{2.16}$$

It should be noted that eq. 2.15 is the same for one-photon or two-photon excitations, but the diffusion time of eq. 2.16 is half that of one-photon excitation, because the effective size of the beam focus is $\frac{w_o}{\overline{2}}$.

2.3.2 Gaussian-Lorentzian Model of Intensity and Correlation Functions

The Gaussian-Lorentzian model of the laser focus has a Gaussian distribution of intensity in the optical plane (*x*-axis and *y*-axis) and a Lorentzian distribution along the optical axis (*z*-axis)[7] and is represented as

$$I\ x,y,z\ = \frac{2I_o w_o^2}{\pi w^2(z)} \exp\ \frac{-2\ \ x-x_0\ ^2+\ y-y_0\ ^2}{w^2(z)}\ , \tag{2.17}$$

where

$$w^2\ z\ = w_o^2\ \ 1+\ \frac{z-z_0}{z_R}\ ^2\ , \tag{2.18}$$

and z_R is the Rayligh range. The numerator of the first term of equation 2.5, with $I(r,t)$ replaced with equation 2.17, becomes

$$\delta F(r,t)\delta F(r,t+\tau)\ = \frac{4\alpha^2 I_o^4 w_o^4\ C}{(4\pi D\tau)^3\ {}_2\pi^2}\quad dx'dy'dz'\quad dx''dy''dz''$$

$$\times \frac{1}{w^4\ z'\ w^4\ z''} \exp\ \frac{-4\ x'-x_\iota\ ^2}{w^2\ z'}\ \exp\ \frac{-4\ y'-y_\iota\ ^2}{w^2\ z'}$$

$$\times \exp\ \frac{-4\ x''-x_J\ ^2}{w^2\ z''}\ \exp\ \frac{-4\ y''-y_J\ ^2}{w^2\ z''}$$

$$\times \exp\ \frac{-(\ x''-x'-v_x\tau\ ^2+\ y''-y'-v_y\tau\ ^2+\ z''-z'-v_z\tau\ ^2)}{4D\tau}\ . \tag{2.19}$$

The Gaussians in equation 2.19 can be combined and simplified to obtain the two-photon cross-correlation equation for $g_{ij}(\tau)$:

$$g_{ij}(\tau) = \frac{\alpha^2 I_o^2 w_o^4 C}{8\pi\sqrt{4\pi D\tau}} \int dz' \int dz'' \frac{1}{w^2(z')\, w^2(z'')}$$

$$\times \frac{1}{2D\tau + \frac{w^2(z')}{8} + \frac{w^2(z'')}{8}} \exp\left[\frac{-(x_j - x_i - v_x\tau)^2}{4D\tau + \frac{w^2(z')}{4} + \frac{w^2(z'')}{4}}\right]$$

$$\times \exp\left[\frac{-(y_j - y_i - v_y\tau)^2}{4D\tau + \frac{w^2(z')}{4} + \frac{w^2(z'')}{4}}\right] \exp\left[\frac{-(z'' - z' - v_z\tau)^2}{4D\tau}\right] + 1, \qquad 2.20$$

where

$$w^2(z') = w_o^2\left[1 + \left(\frac{z' - z_i}{z_R}\right)^2\right] \qquad 2.21$$

and

$$w^2(z'') = w_o^2\left[1 + \left(\frac{z'' - z_j}{z_R}\right)^2\right]. \qquad 2.22$$

This form of the correlation function does not have an analytical solution, and hence numerical integration must be used when applying it to curve fir experimental correlation functions. A similar approach was taken for fitting the autocorrelation function in the first published account of two-photon FCS,[7] but here the equation is generalized to account for cross-correlation functions and flow between two separated foci.

CHAPTER III-EXPERIMENTAL APPARATUS

3.1 Initial Two-Photon Four-Focus Microscope Measurements

Four foci arranged in a tetrahedral pattern can be used as a 3D standard ruler for measuring flow velocity in solution. The foci are placed such that a pair of foci form one axis of observation, a second pair form a perpendicular axis to the first, and the combined foci of each axis form a third axis perpendicular to both of the others, as seen in Figure 1 and Figure 2. Experimentally, this is achieved by splitting a single laser beam into four with an optical setup akin to a double Mach-Zehnder [36] interferometer with appropriate delay arms, as seen in Figure 3 and Figure 4. The optical setup also serves to split one pulse from the Ti-Sapphire laser into four pulses, each with different arrival times. Based on the 76 MHz repetition rate of the Ti-Sapphire, the delay between arriving pulses is separated by 3.3 ns; that is, after the first pulse (Focus 1) arrives, the second pulse (Focus 2) arrives 3.3 ns later, the third pulse (Focus 3) arrives 6.6 ns after the first, and the fourth pulse (Focus 4) arrives 9.9 ns after the first pulse. The first pulse of the next set arrives 3.3 ns after the pulse for Focus 4 arrives, that is, 13.2 ns after the arrival of the previous first pulse and starts the cycle over again. Fluorescence is then separable by focus through the use of time-correlated single-photon counting methods and FCCS measurements are done between all four foci. All 16 correlation curves (4 autocorrelation curves and 12 cross-correlation curves) are dependent on the global velocities present in the sample.

Preliminary FCCS experiments were performed with a 144 pM aqueous solution of Rhodamine B flowing through a square-bore capillary. Autocorrelation functions were constructed from the collected fluorescence, as seen in Figure 5. Based on the peaks of the autocorrelation function, it was determined that the signal-to-noise ratio was not large enough to

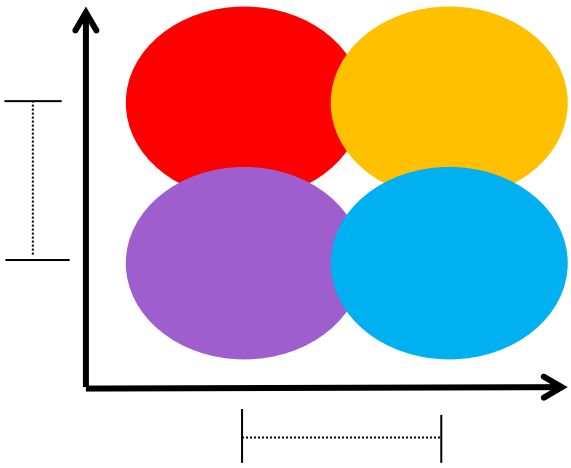


Figure 1. Four foci arranged in a tetrahedral pettern and viewed along the optical axis with the positive z direction coming out of the page. Separation between foci along the x-axis and y-axis is set at 500 nm.

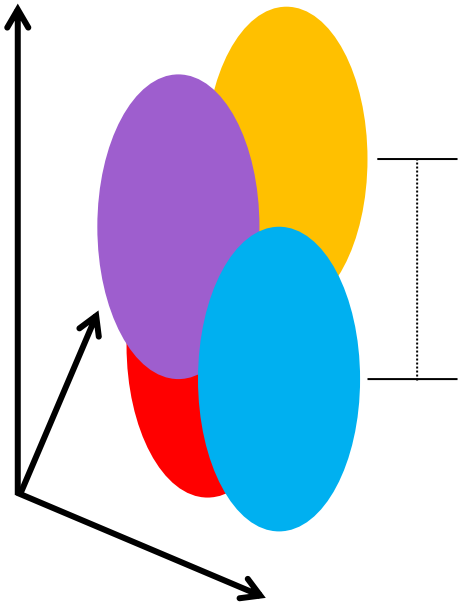


Figure 2. Four foci arranged in a tetrahedral pattern. The distance between the two sets of foci along he optical axis (1 and 2 on the bottom and 2 and 3 on top) is 980 nm.

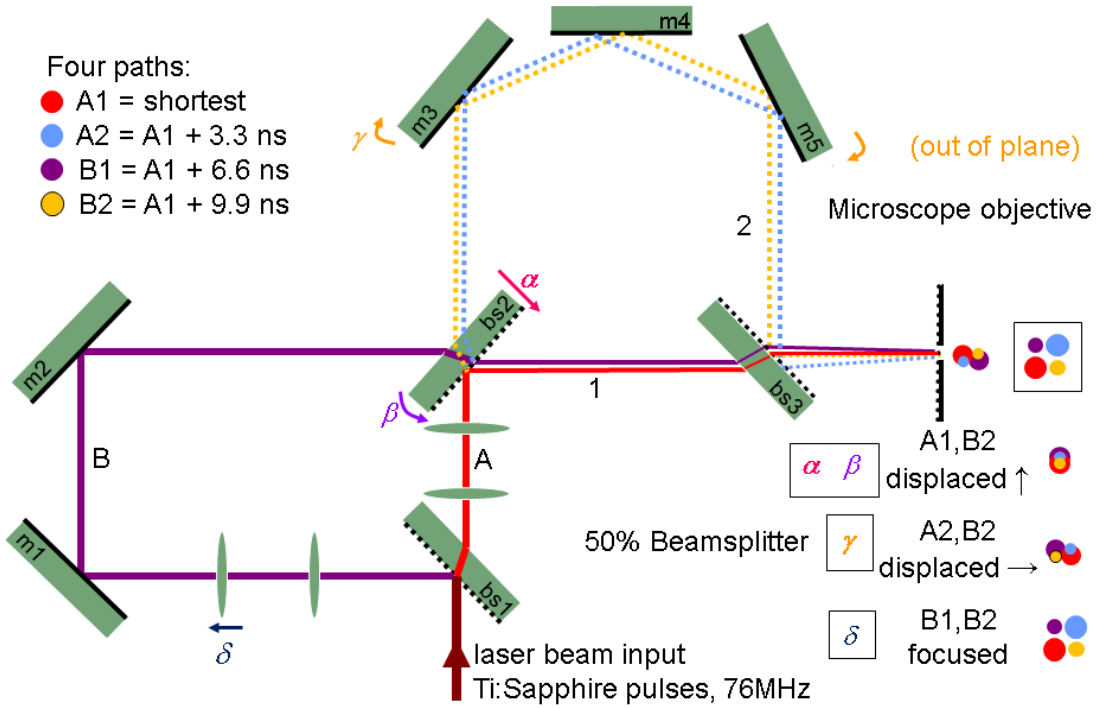


Figure 3. Beam Splitters, mirrors and lenses for spatially and temporally separating a pulsed laser source. The laser source used here is a Ti-Sapphire laser with repetition rate of 76 MHz and wavelength of 800 nm. The beam splitters reflect 50% of each pulse and the length of each arm adds a delay to the reflected pulse. A pulse arrives at the microscope objective every 3.3 ns, with a new series of pulses arriving every 13.2 ns. The focus of each path is spatially separated by rotating and translating beam splitter 2 (bs2), rotating the mirrors of arm two (m3, m4, and m5) and varying the distance between the lenses in arm B.

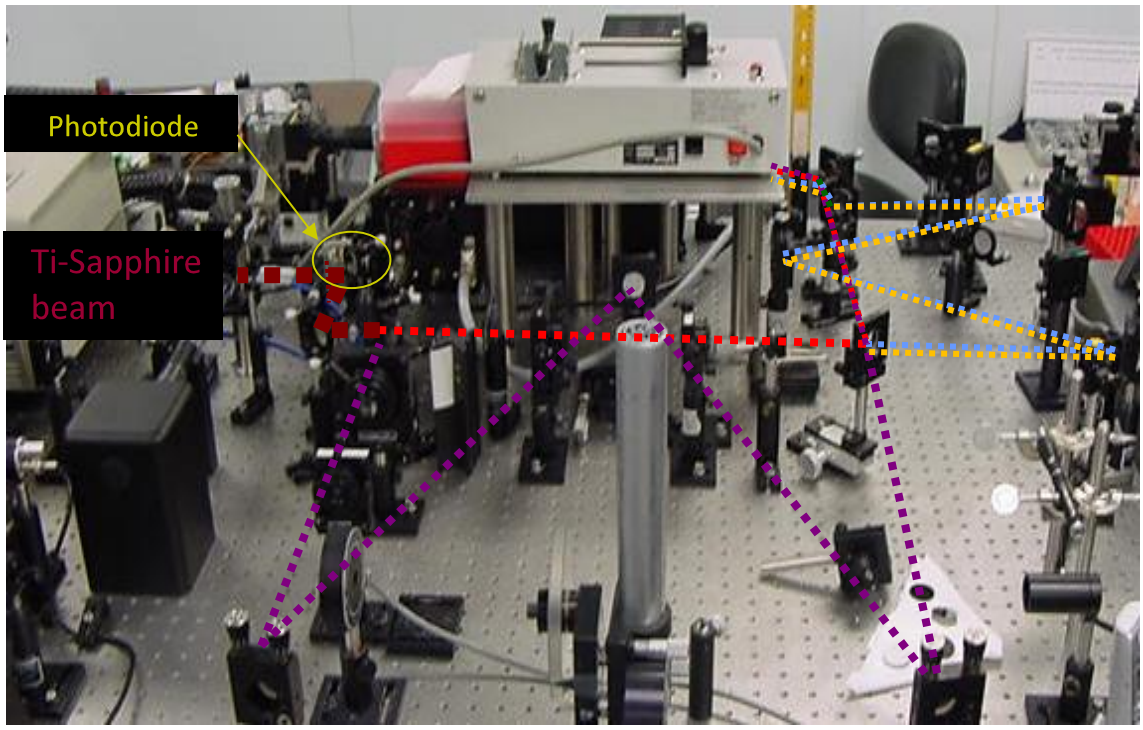


Figure 4. Two-photon four-focus microscope. Excitation light is provided by a 76 Mhz Ti-Sapphire pulsed laser, wavelength 800 nm. The differing length between beam paths creates a temporal separation in multiples of 3.3 ns. Spatial separation is created by adjusting the beam splitters and mirrors. The photodiode is used to signal to start a new counting cycle for time-resolved measurements.

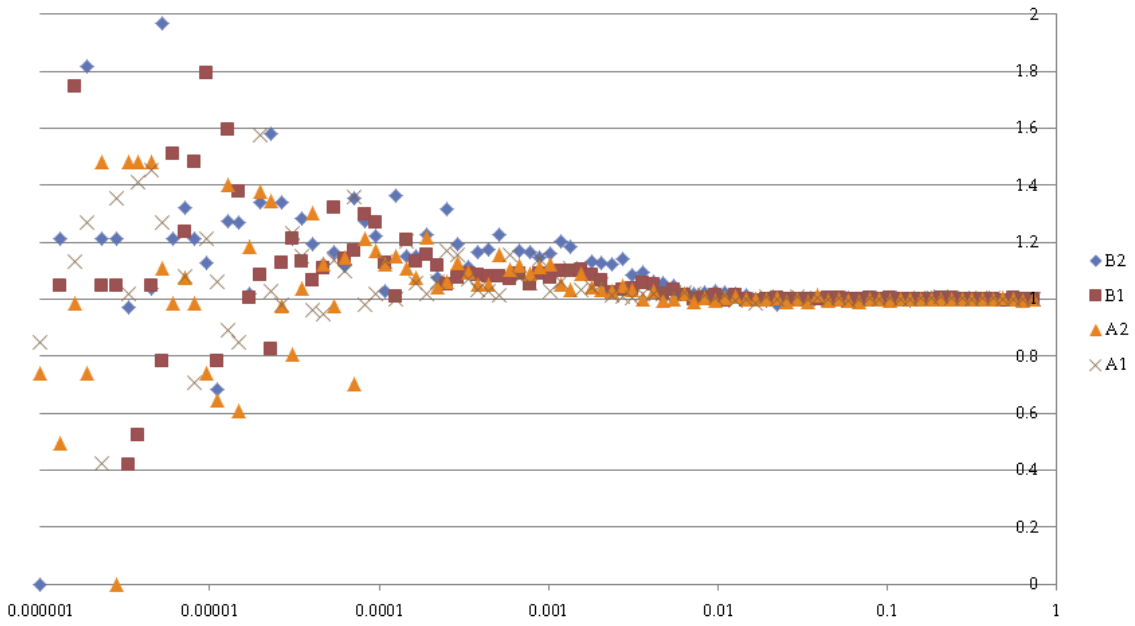


Figure 5. Autocorrelation functions of initial FCCS experiments with two-photon, four-focus microscope. The low autocorrelation peaks reflects a low signal-to-noise ratio.

allow for accurate FCCS measurements. The poor signal-to-noise was in part due to dispersion of the femtosecond laser pulses in passing through the beam splitters and the microscope objective. Due to failure of a pump laser, the experiment had to be rebuilt using a different Ti-Sapphire laser, which did not have a means for dispersion compensation. While awaiting repair of the first Ti-Sapphire laser system (which did have dispersion compensation), simulations of flow through the four-focus system were conducted. The goal of these simulations was to determine the accuracy with which components of 3D flow could be measured by curve-fitting FCCS measurements, for realistic estimates of experimental background levels and photon statistics.

3.2 Two-Photon Simulations

An in-house simulation, based on earlier simulations,[37] was created at UTSI that simulates particles moving through a four-focus tetrahedral region. As the beams would not fill an objective in an actual experiment, the focal volumes were represented by Gaussian-Lorentzian intensity profiles.[38] A number of virtual particles were distributed throughout a test region based on the simulated concentration. The particles took random diffusional steps in the volume and also undertook directed steps based on simulated flow parameters. When a simulated particle entered into one of the excitation volumes, there was a probability with each time step that the particle absorbed a photon. As the process under study was a two-photon process, two photons had to be absorbed to generate fluorescence. The number of photons available for absorption was based on the average irradiance of each beam at the location of the particle. When the particle entered an excited state, four outcomes were possible: (i) The particle entered a singlet state (fluorescent state) and subsequently decayed to the ground state after a random

time determined by the fluorescence lifetime, whereupon a photon was detected; (ii) The particle entered a triplet state (phosphorescent state) and subsequently returned to the ground state after a random time with mean equal to the phosphorescence lifetime; (iii) The particle was photobleached; or (iv) The particle returned to the ground state after a random time equal to the fluorescence lifetime but without detection of a photon (the excitation decayed non-radiatively or radiatively but without detection of a photon). Fluorescence generated by the particle was then collected and time-stamped on arrival. The photon counts generated by the simulation were then analyzed with a custom LabVIEW virtual instrument (VI) that calculates the 16 correlation functions and builds a spreadsheet text file that contains the correlation function at time τ.

A Python script (Appendix A) was constructed to analyze the data exported by the LabVIEW correlator VI. The script imports the time delays, the sixteen correlation functions, and the number of photon pairs used to calculate each point from the spreadsheet file. In order to curve fit the sixteen correlation functions to obtain the flow velocity components, the script begins by computing the sixteen curves using either a 3D Gaussian approximation or a Gaussian-Lorentzian model. In the Gaussian-Lorentzian case, the integration over the optical axis (the z' and z'' variables in equation 2.19) is done numerically with a Simpson's rule routine developed and implemented in C. This C code is called in-line from the Python script. These original cross-correlation functions do not account for signal crosstalk, where photons generated by one focus are counted in another focus' bin, as seen in Figure 6. The probability of a photon being incorrectly attributed to a different focus can be calculated from the fluorescence lifetime, τ_f, and the arrival time difference between the correct pulse and the incorrect pulse. This probability is expressed as[36]

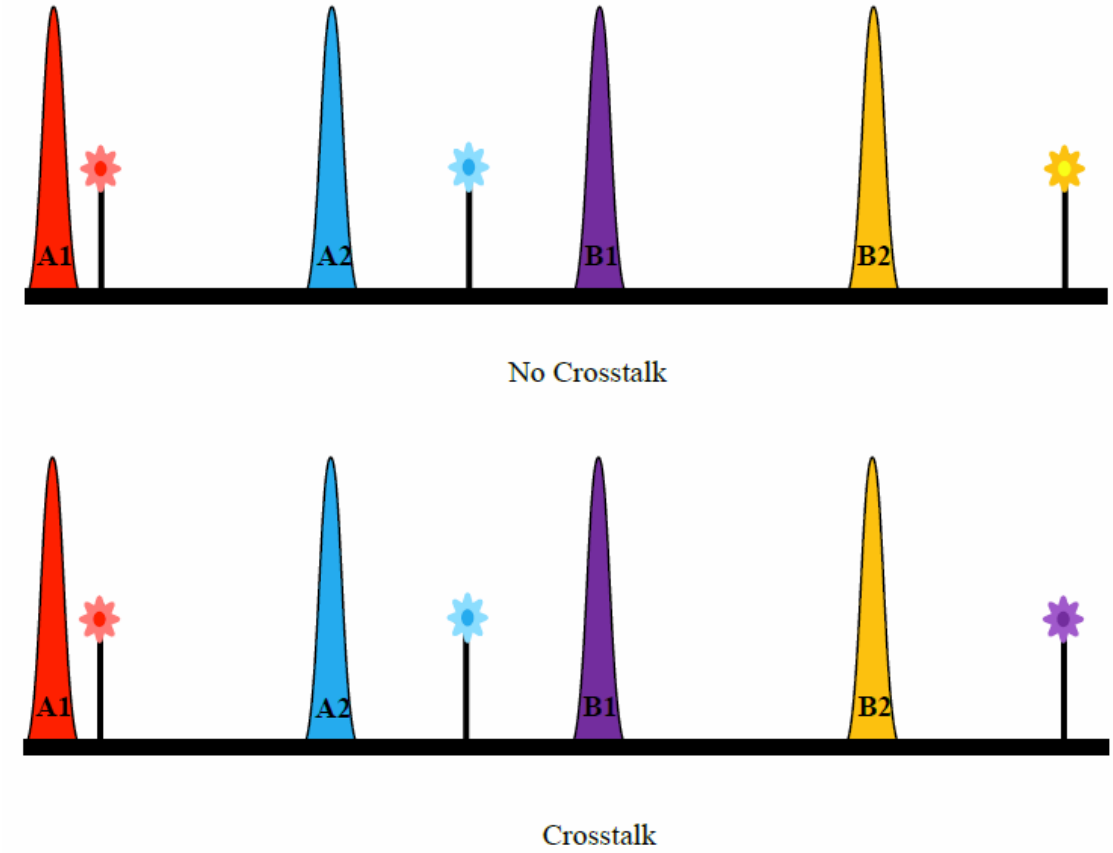


Figure 6. Examples of crosstalk and no crosstalk counting. In no crosstalk counting all photons are assumed to be generated by the pulse that has just arrived at the sample. In crosstalk counting there is a finite probability that a photon generated from one laser pulse can be incorrectly associated with another pulse. The probability is based on the lifetime of the fluorescent species and the difference between the arrival times of pulses.

$$p_{ij} = \exp\ \frac{-(n_{ij}) * 3.3\ ns}{\tau_f}\ - \exp\ \frac{-(3.3\ ns *\ \ n_{ij} + 1\ )}{\tau_f}\ , \qquad 3.1$$

where p_{ij} is the probability of a photon generated by a pulse being counted correctly, *n* corresponds to the bin number after the pulse has arrived (that is $n_{ij} = j - i$, with *i* as the bin generating the photon and *j* as the bin the photon is counted in with values of *j* are restricted to *i* + 0, *i* + 1, *i* + 2, *i* +3). The fluorescence lifetime is chosen to be 1.8 ns, which is the approximate lifetime of Rhodamine B.[39] The lifetime of Rhodamine B is sufficiently small that over 80% of generated photons are counted correctly, and it is easily excitable at 800 nm.[20] The probabilities of crosstalk between the foci are listed in Table 1. The final correlation functions, $g_{ij}(\tau)$, returned by the Python script are computed by summing all the correlation functions multiplied by the probabilities of each photon being generated by a particular focus, i.e.,

$$g_{ij}\ \tau\ =\ \ \sum_{l=0}^{4}\ \ \sum_{m=0}^{4} p_{il} p_{mj} g_{lm}(\tau). \qquad 3.2$$

A total of 19 different adjustable variables were used to fit the model equations to the correlation curves obtained from the simulations. Values of some parameters, such as the beam waist, were held fixed, as these could be obtained from calibration experiments. Three adjustable parameters were used for the flow velocity. Also, the amplitudes of the 16 correlation functions were allowed to be separately adjustable, although they are related to each other through the cross-term coefficients in eqn. 3.2 and the prefactor term in eqn. 2.20. The built-in least-squares fitter (leastsq) package from Python was used to perform the fitting.

Table 1. The probability of photons counted in each bin for four foci. The row corresponds to the originating focus and the column corresponds to the focus for which the photon is counted.

p_{ij}	0	1	2	3
0	0.840	0.134	0.021	0.003
1	0.003	0.840	0.134	0.021
2	0.021	0.003	0.840	0.134
3	0.134	0.021	0.003	0.840

3.2 Two-Photon Fitting: Variations to Simulations

To test if the 3D Gaussian model could accurately estimate flow or if the Gaussian-Lorentzian model was required to provide an accurate fit, the simulation was run for various cases of experimental set-ups and for different values of flow velocities. The first case tested was to have only one focus "on", with an average irradiance of 5 mW (66pJ/pulse). The simulated average power for all foci besides the "on" focus was turned down to essentially zero (while avoiding a divide by zero error for a setting of zero). Simulation times were set for ten seconds.

The second case tested was with just two foci on. The first focus and the focus that arrives 6.6 ns after the first focus were set to have an average of 5 mW of power each. The remaining two foci were set to essentially zero power. Spatially, these foci are located on the same optical plane and separated by a distance of 500 nm. Crosstalk for two foci is accounted for and is tabulated in Table 2. Flow is simulated to run parallel to the line connecting the centers of the two foci.

Simulations with all foci set to the "on" state were then run. Flow was directed first along one of the axes in the optical plane, and then along the optical axis. Finally, a flow

directed along multiple axes was simulated. Simulations with the triplet state both enabled and disabled were performed.

Table 2. The probability of photons counted in each bin for two foci. The row corresponds to the originating focus and the column corresponds to the focus for which the photon is counted.

p_{ij}	0	1
0	0.974	0.026
1	0.026	0.974

CHAPTER IV-DATA OF TWO-PHOTON FCCS FITTING

4.1 One-Focus Simulation Fitting

Initial simulations began with one focus and one-dimensional flow. Simulated flow speeds, fitted flow speeds, and the differences between the fitted and simulated values are shown in Table 3 for triplet state fitting and Table 4 for fitting without triplet state. Differences between the simulated and fitted values for each of the two models are displayed in Figure 7. Examples of correlation fits are shown in Figure 8 and Figure 9. The average absolute difference in fitting for the triplet state is 10.9 mm/s for the 3D Gaussian case and in the Gaussian-Lorentzian case is 0.7 mm/s. The average absolute difference for fitting with the triplet state is 8.9 mm/s for the 3D Gaussian case and 0.7 mm/s in the case of the Gaussian-Lorentzian. The simulations for one-focus fitting show that the results from the Gaussian-Lorentzian model consistently have a lower absolute difference between fit velocities and simulated flow velocity than that for the 3D Gaussian model. The differences between the Gaussian-Lorentzian model with and without fitting for the triplet state is small (<0.2 mm/s). For the 3D Gaussian, triplet fitting did not improve flow recovery. Based on the result that triplet fitting did not significantly decrease the difference between simulated and fitted flow further fittings did not include triplet effects.

Table 3. Simulated and fitted flow speeds for 3D Gaussian and Gaussian-Lorentzian fitting models taking into account the triplet state parameters. The difference between simulated and fitted flow velocity for each model is listed next to each flow fit value.

Sim. Flow (mm/s)	3D Gaussian (mm/s)	Difference	Gaussian-Lorentzian (mm/s)	Difference
0	0	0.0	0	0.0
10	12.5	-2.5	9.7	0.3
20	25.9	-5.9	20.1	-0.1
50	61.0	-11.0	47.8	2.2
100	135.0	-35.0	99.1	0.9

Table 4. Simulated and fitted flow speeds for 3D Gaussian and Gaussian-Lorentzian fitting models without triplet state parameters. The difference between simulated and fitted flow for each model is listed next to each fit flow value.

Sim. Flow (mm/s)	3D Gaussian (mm/s)	Difference	Gaussian-Lorentzian (mm/s)	Difference
0	0	0.0	0	0.0
10	8.9	1.1	9.7	0.3
20	25.8	-5.8	20.1	-0.1
50	61.0	-11.0	47.6	2.4
100	126.4	-26.4	99.1	0.9

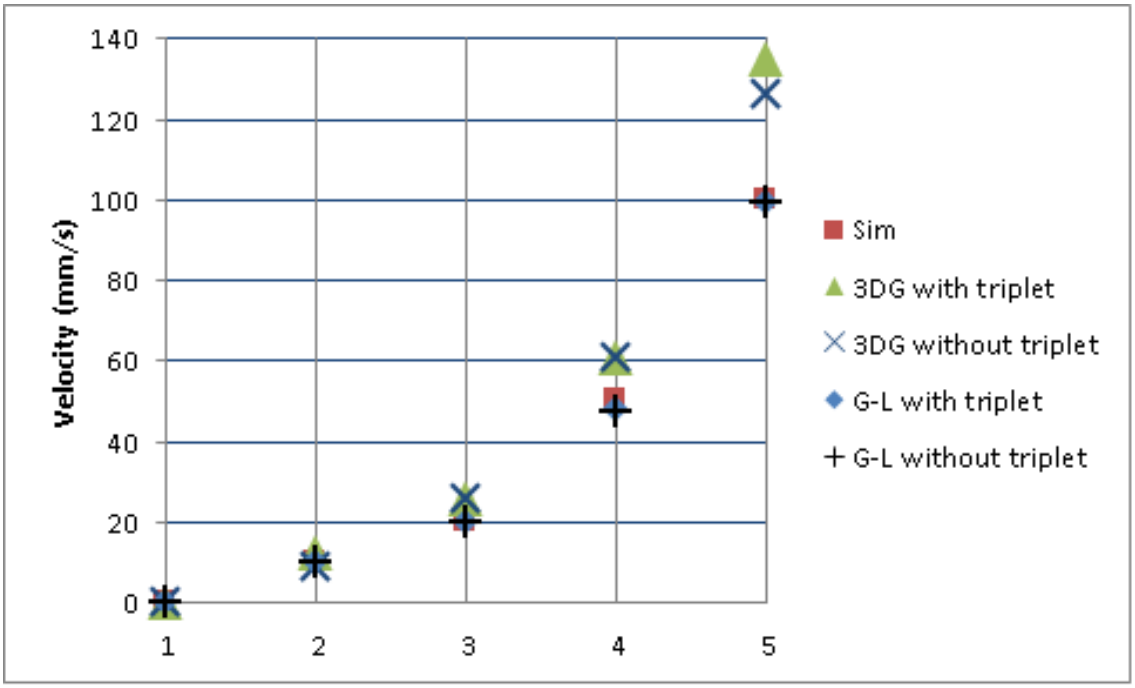


Figure 7. Simulated and recovered flow velocities. Run velocity are; (1) 0 mm/s, (2) 10 mm/s, (3) 20 mm/s, (4) 50 mm/s, and (5) 100 mm/s.

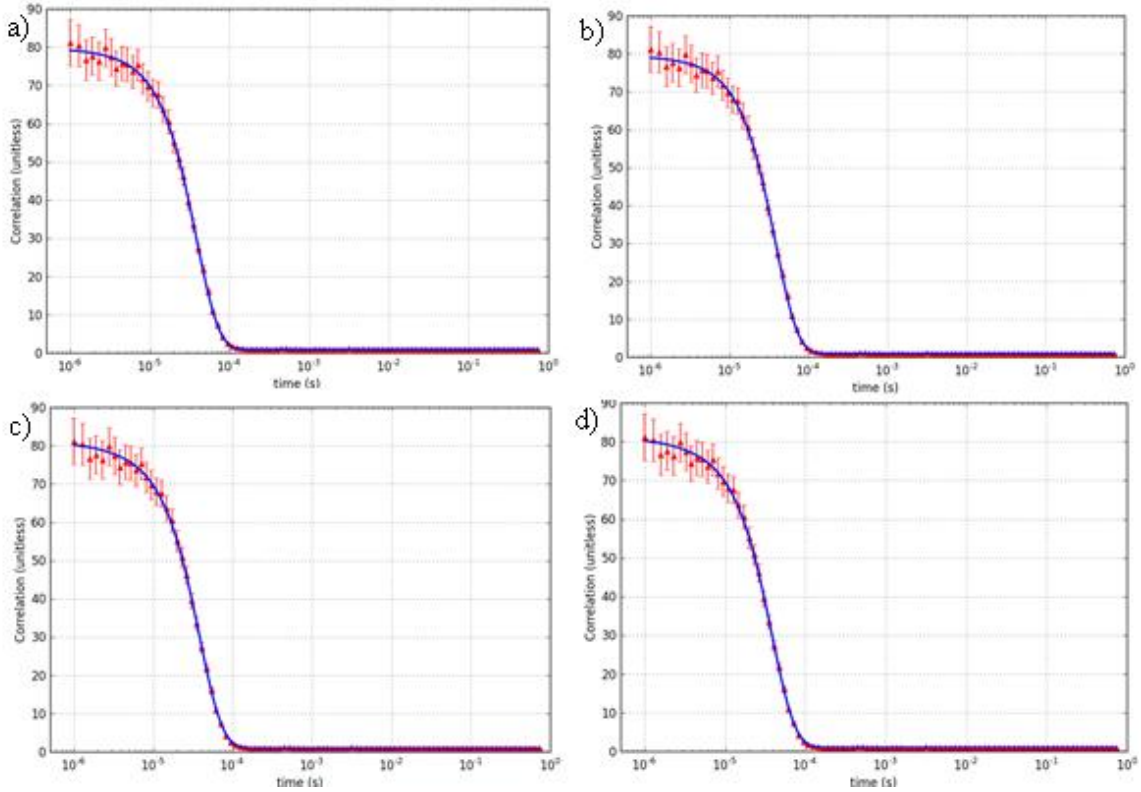


Figure 8. Simulated autocorrelation fluorescence and fitting with the model; a) 3D Gaussian with triplet fitting; b) 3D Gaussian without triplet fitting; c) Gaussian-Lorentzian with triplet fitting; d) Gaussian-Lorentzian without triplet fitting. The blue line is the fit and the red symbols are simulated correlation points. Error bars, which are determined from the number of photon pairs that contribute to the correlation, are used for weighting the fitting. The simulated flow speed was 10 mm/s and the fits were a) 12.5 mm/s, b) 8.9 mm/s, c) 9.7 mm/s, and d) 9.7 mm/s.

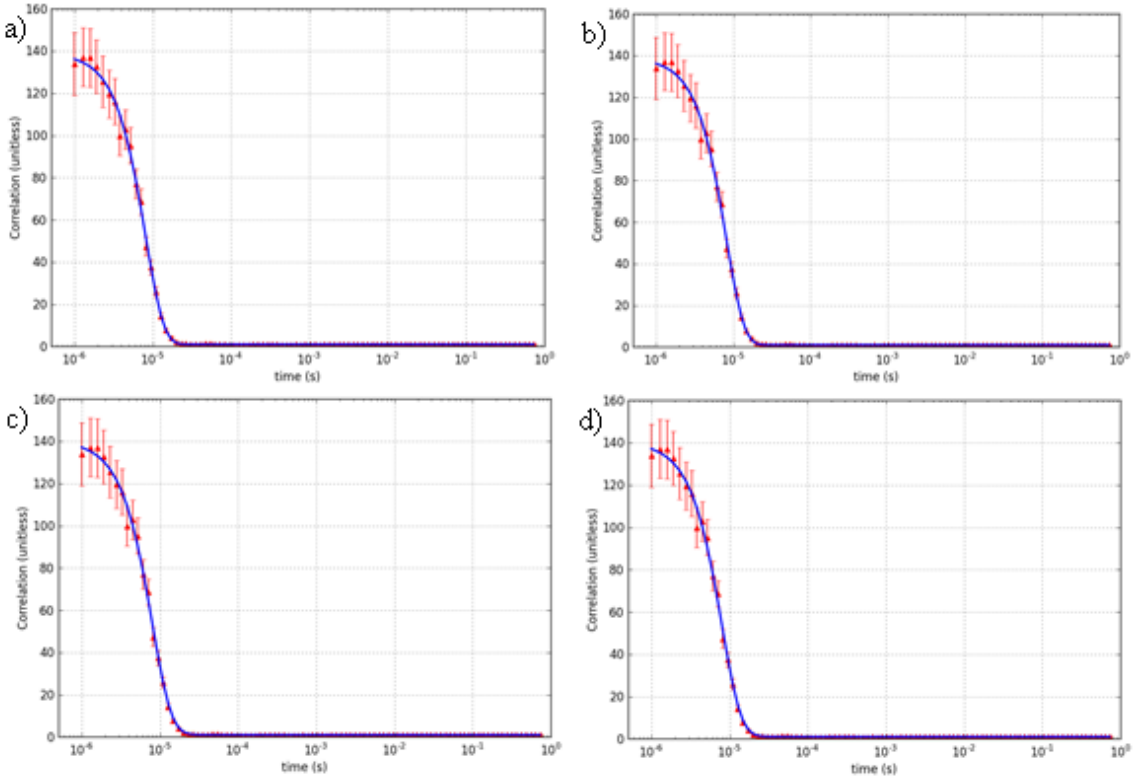


Figure 9. Simulated autocorrelation fluorescence and fitting with the model; a) 3D Gaussian with triplet fitting; b) 3D Gaussian without triplet fitting; c) Gaussian-Lorentzian with triplet fitting; d) Gaussian-Lorentzian without triplet fitting. The blue line is the fit and the red symbols are simulated correlation points. Error bars, which are determined from the number of photon pairs that contribute to the correlation, are used for weighting the fitting. The simulated flow speed was 50 mm/s and the fits were a) 61.0 mm/s, b) 61.0 mm/s, c) 47.8 mm/s, and d) 47.6 mm/s.

4.2 Two-focus Simulation Fitting

Simulations for two foci were conducted with flow parallel to the *x*-axis. Fits of the correlations curves were conducted using both the 3D Gaussian model and the Gaussian-Lorentzian model. Simulated flow speeds, fitted flow speeds, and differences between the fitted values and simulated values are noted in Table 5. Simulated and fitted values are plotted in Figure 10. Figure 11 and Figure 12 show examples of 3D Gaussian and Gaussian-Lorentzian fits. The average absolute difference is 2.1 mm/s for the 3D Gaussian fits and 0.3 mm/s for the Gaussian-Lorentzian fits. In the case of one-dimensional flow with two foci, the Gaussian-Lorentzian function proves to be more accurate in finding the simulation flow speed.

Table 5. Simulated and fitted flow speeds for 3D Gaussian and Gaussian-Lorentzian fitting models. The fractional error for each model is listed next to each flow fit value. The values for the Gaussian-Lorentzian model have a lower absolute difference between fitted and simulated flow velocities in all cases.

Sim. Flow (mm/s)	3D Gaussian (mm/s)	Difference	Gaussian-Lorentzian (mm/s)	Difference
5	6.1	-1.1	4.8	0.2
10	11.9	-1.9	9.7	0.3
20	23.4	-3.4	19.7	0.3

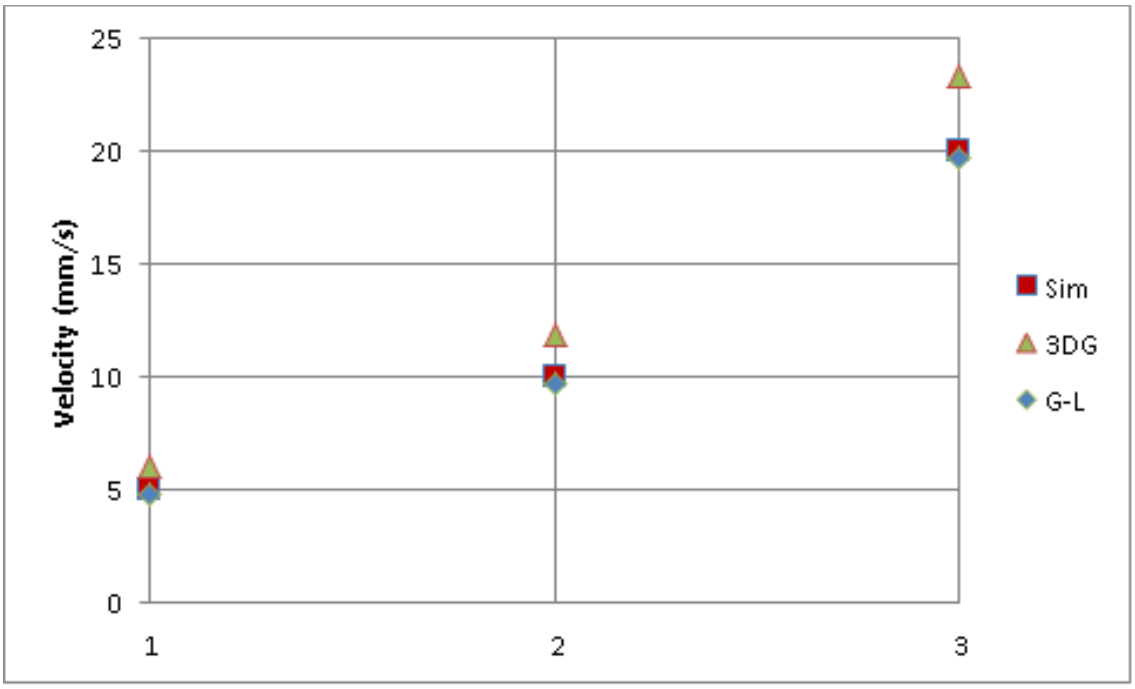


Figure 10. Simulated and recovered flow velocities with simulated velocities set at; (1) 5 mm/s, (2) 10 mm/s, and (3) 20 mm/s. 3D Gaussian and Gaussian-Lorentzian models with one-dimensional flow parallel to the axis of displacement between the foci. The Gaussian-Lorentzian model in this case proved to be more accurate in all three test cases.

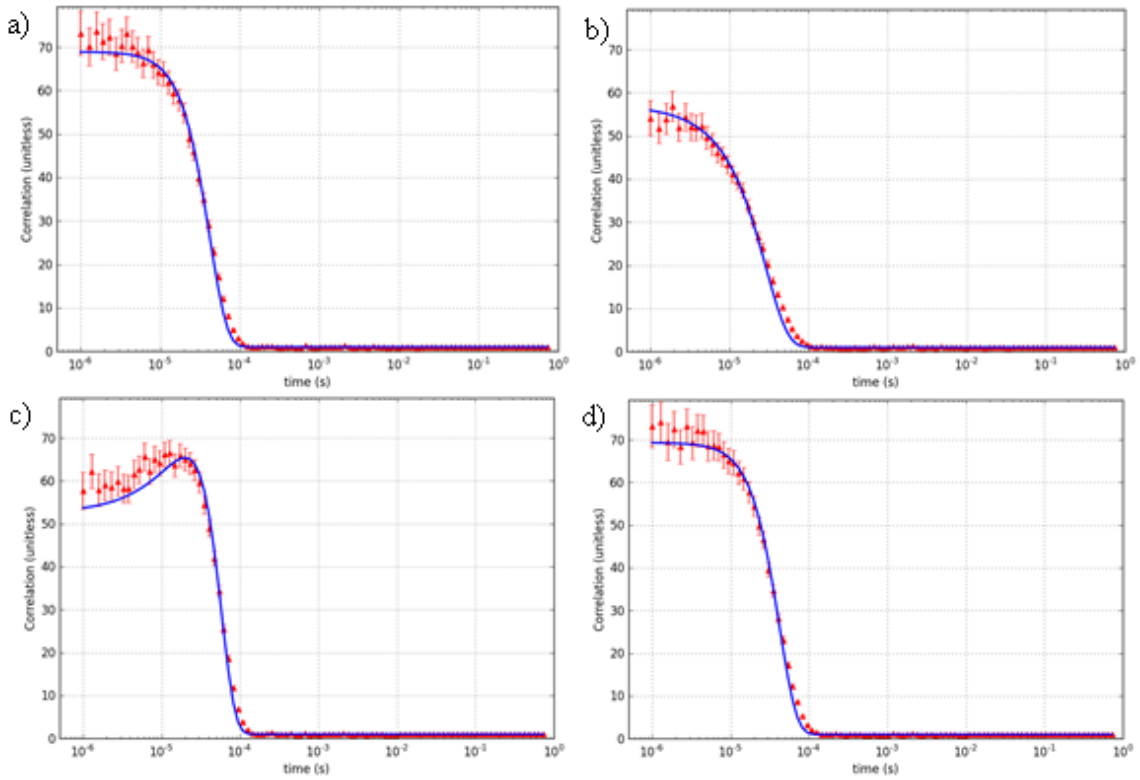


Figure 11. 3D Gaussian fits for 10 mm/s flow with two foci: a) autocorrelation of focus 1; b) cross-correlation from focus 1 to focus 2; c) cross-correlation from focus 2 to focus 1; d) autocorrelation of focus 2. Flow was parallel to the displacement between foci. The blue lines are fits and the red symbols are the simulated correlation points. Error bars, which are determined from the number of photon pairs that contribute to the correlation, are used for weighting the fitting. The flow speed determined from the fit was 11.9 mm/s.

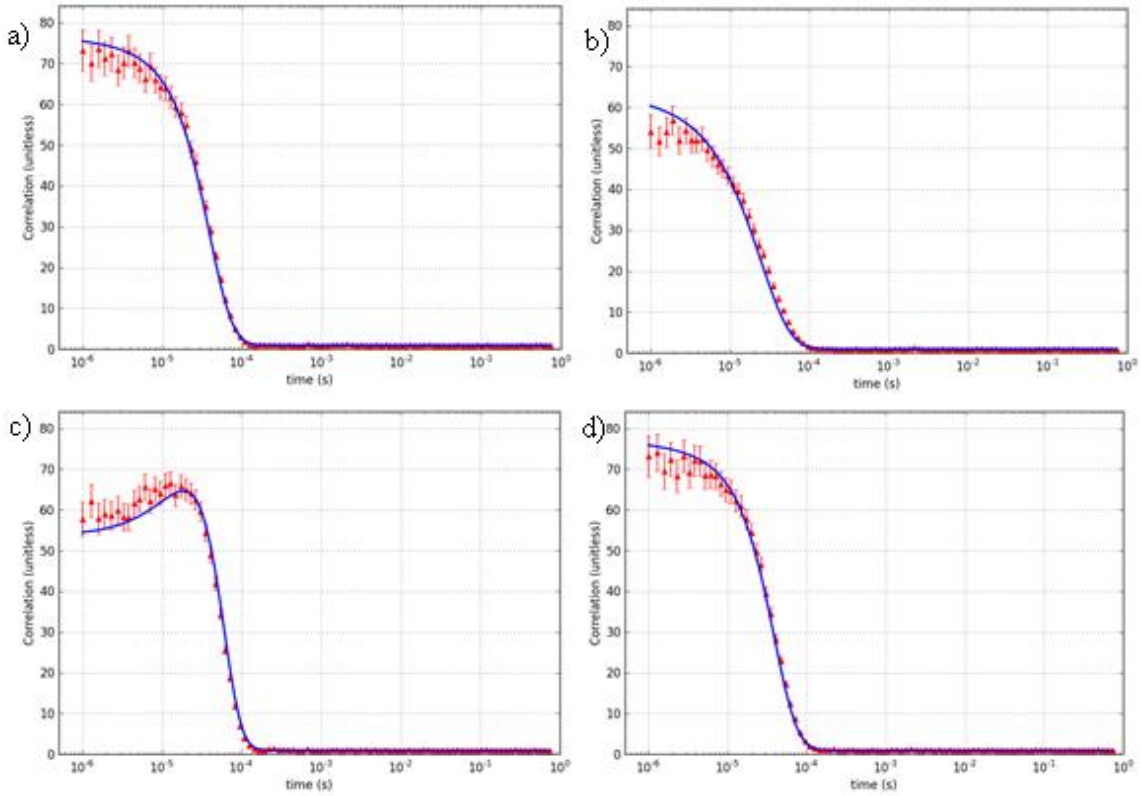


Figure 12. Gaussian-Lorentzian fits for 10 mm/s flow with two foci: a) autocorrelation of focus 1; b) cross-correlation from focus 1 to focus 2; c) cross-correlation from focus 2 to focus 1; d) autocorrelation of focus 2. Flow was parallel to the displacement between foci. The blue lines are fits and the red symbols are the simulated correlation points. Error bars, which are determined from the number of photon pairs that contribute to the correlation, are used for weighting the fitting. The flow speed determined from the fit was 9.7 mm/s.

4.3 Four-focus Simulation Fitting

Six simulations of 3D flow through all four foci were conducted. The simulated (x, y, z) flow velocities, fitted flow velocities, and differences are recorded in Table 6. A comparison between the fitted values of the flow velocity components and the simulated ones is shown in Figure 13. An example of 3D Gaussian fitting and Gaussian-Lorentzian fitting for the case of freely diffusing particles—v_x = 10.0 mm/s, v_y = 10.0 mm/s, and v_z = 20.0 mm/s—is shown in Figure 14 and Figure 15. From Figure 13, the simulated velocities can be approximated and the trend of fitted velocities follows the trend of simulated velocities along the x-axis and y-axis. However, the simulated and fitted velocities along the z-axis have substantial error. The average absolute differences for the 3D Gaussian model were 2.1 mm/s along the x-axis, 1.8 mm/s along the y-axis, and 6.9 mm/s along the z-axis. For the Gaussian-Lorentzian model, the average absolute differences were 2.1 mm/s along the x-axis, 1.2 mm/s along the y-axis, and 8.5 mm/s along the z-axis. On average, the Gaussian-Lorentzian model is more accurate along the optical plane but less accurate along the optical axis than the 3D Gaussian model.

Table 6. Simulated and fitted (*x*, *y*, *z*) flow velocities with fractional errors for 3D Gaussian and Gaussian-Lorentzian fitting models in a four-focus simulation. Differences between each axis are listed next to each fit value. Neither the 3D Gaussian nor the Gaussian-Lorentzian model has a distinct advantage in this more complicated case.

Simulated Flow (mm/s)	3D Gaussian Fit (mm/s)	Difference	Gaussian-Lorentzian Fit (mm/s)	Difference
(0, 0, 0)	(0.0, 0.0, 0.0)	(0.0, 0,0, 0,0)	(0.0, 0.0, 0.0)	(0.0, 0.0, 0.0)
(10, 0 ,0)	(10.6, –1.7, 1.9)	(-0.6, 1.7, –1.9)	(6.2, –2.1, 11.1)	(3.8, 2.1, -11.1)
(0, 0, 20)	(3.0, 4.6, 4.2)	(–3.0, –4.6, 15.8)	(–0.5, –0.4, 8.5)	(0.5, 0.4, 11.5)
(1, –5, 1)	(0.0, –5.6, 0.2)	(1.0, 0.6, 0.8)	(3.0, –3.9, 5.2)	(–2.0, –1.1, –4.2)
(5, –10, 15)	(11.8, –11.9, 8.5)	(–6.8, 1.9, 6.5)	(9.0, –11.9, 4.8)	(-4.0, 1.9, 10.2)
(10, –10, 20)	(10.9, –12.0, 3.7)	(–0.9, 2.0, 16.3)	(8.0, –11.5, 6.1)	(2.0, 1.5, 13.9)

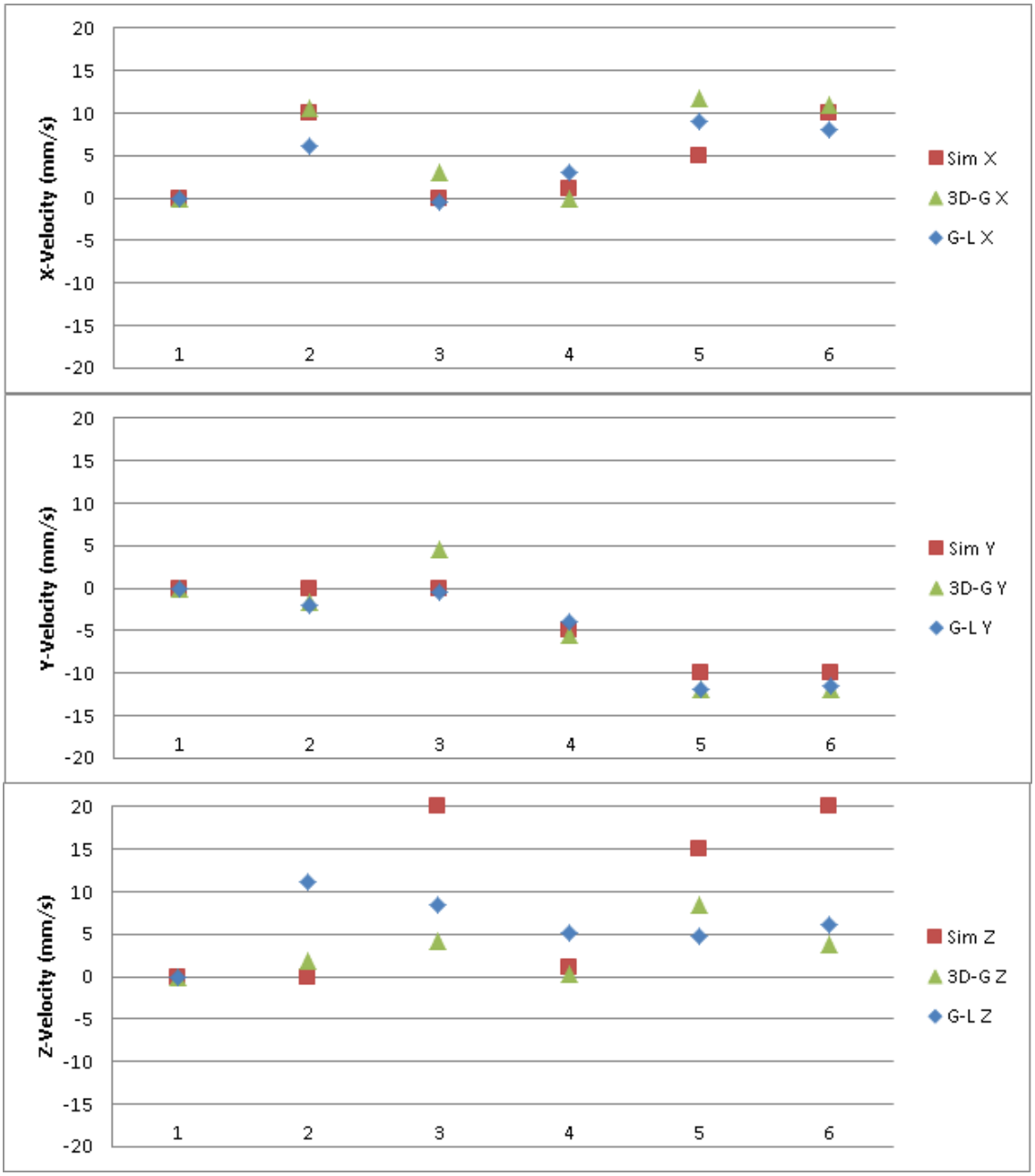


Figure 13. Differences in simulated velocity and fitted velocity along the x, y, and z-axis. Neither the 3D Gaussian nor the Gaussian-Lorentzian is consistently closer to the simulated velocity.

Figure 14. 3D Gaussian fits for flow velocities v_x = 10.0 mm/s, v_y = –10.0 mm/s, v_z = 20.0 mm/s with correlation functions g_{ij}: a) 11; b) 12; c) 13; d) 14; e) 21; f) 22; g) 23; h) 24; i) 31; j) 32; k) 33; l) 34; m) 41; n) 42; o) 43; p) 44. The blue lines are fits and the red symbols are the simulated correlation points. Error bars, which are determined from the number of photon pairs that contribute to the correlation, are used for weighting the fitting. The flow speeds determined from the fit were v_x = 10.9 mm/s, v_y = –12.0 mm/s, v_z = 3.7mm/s.

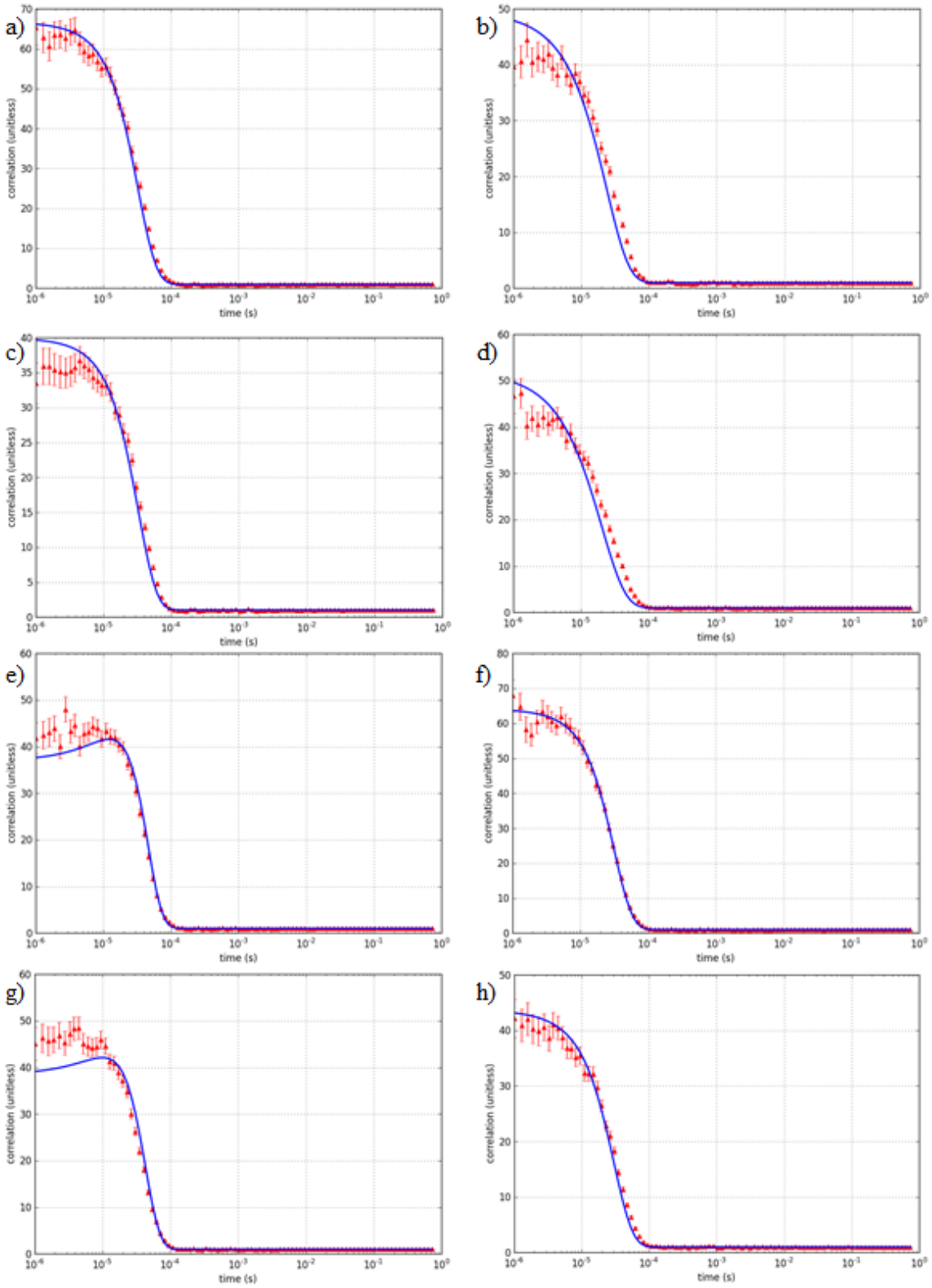


Figure 14. Continued

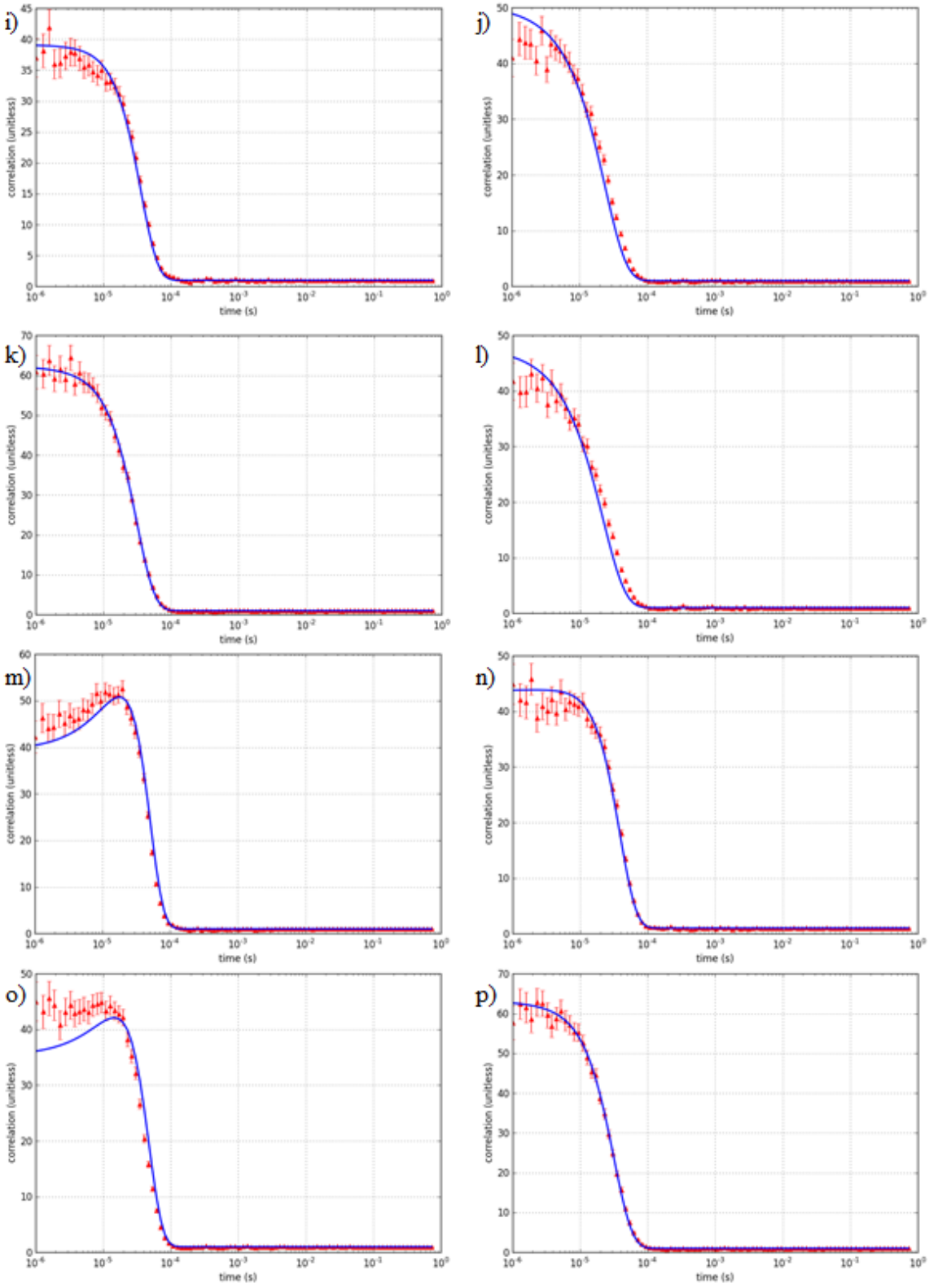

i)
j)
k)
l)
m)
n)
o)
p)
correlation (unitless)
time (s)

Figure 15. Gaussian-Lorentzian fits for flow velocities v_x = 10.0 mm/s, v_y = –10.0 mm/s, v_z = 20.0 mm/s with correlation functions g_{ij}: a) 11; b) 12; c) 13; d) 14; e) 21; f) 22; g) 23; h) 24; i) 31; j) 32; k) 33; l) 34; m) 41; n) 42; o) 43; p) 44. The blue lines are fits and the red symbols are the simulated correlation points. Error bars, which are determined from the number of photon pairs that contribute to the correlation, are used for weighting the fitting. The flow speeds determined from the fit were v_x = 8.0 mm/s, v_y = –11.5 mm/s, v_z = 6.1mm/s.

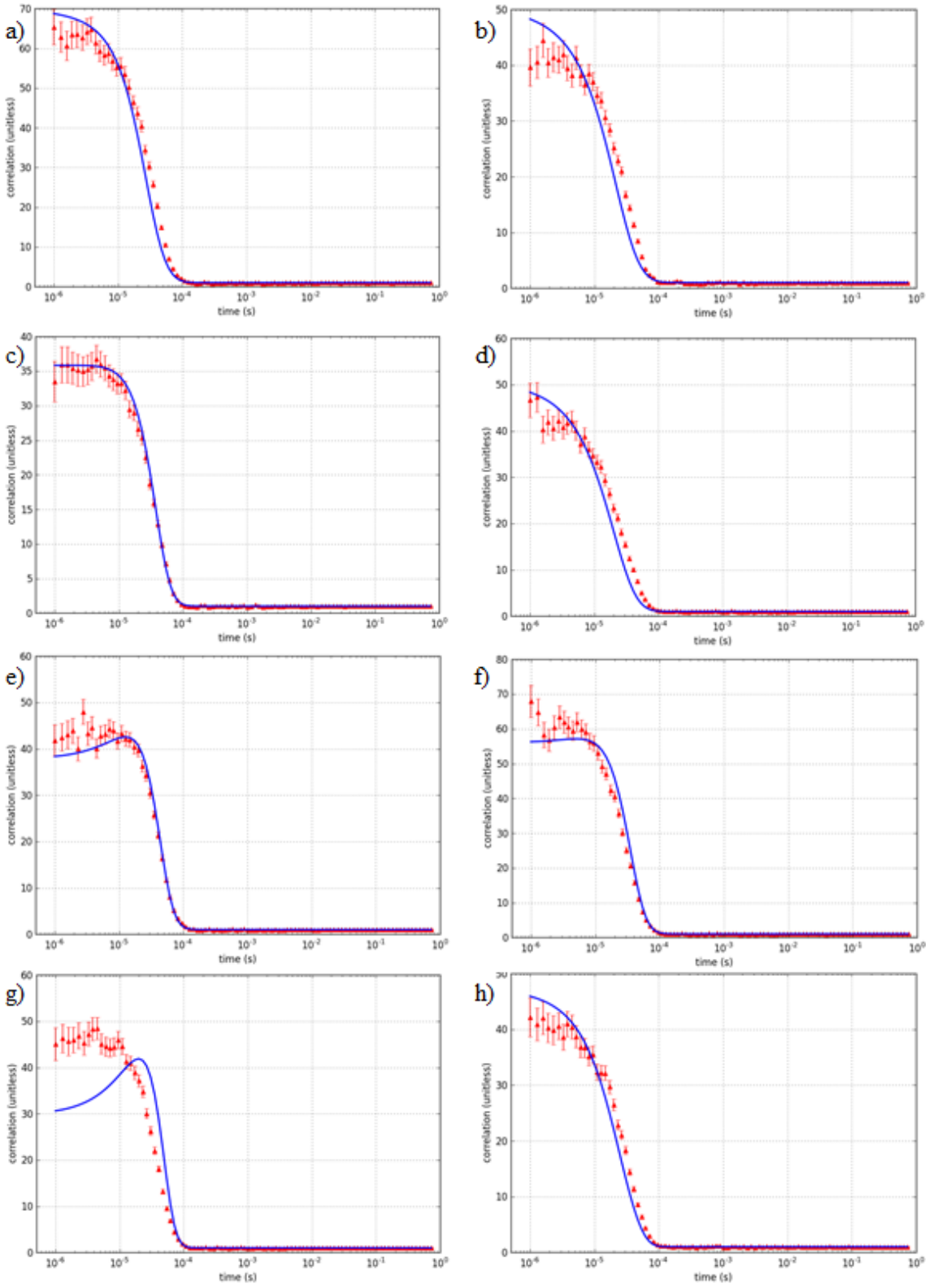
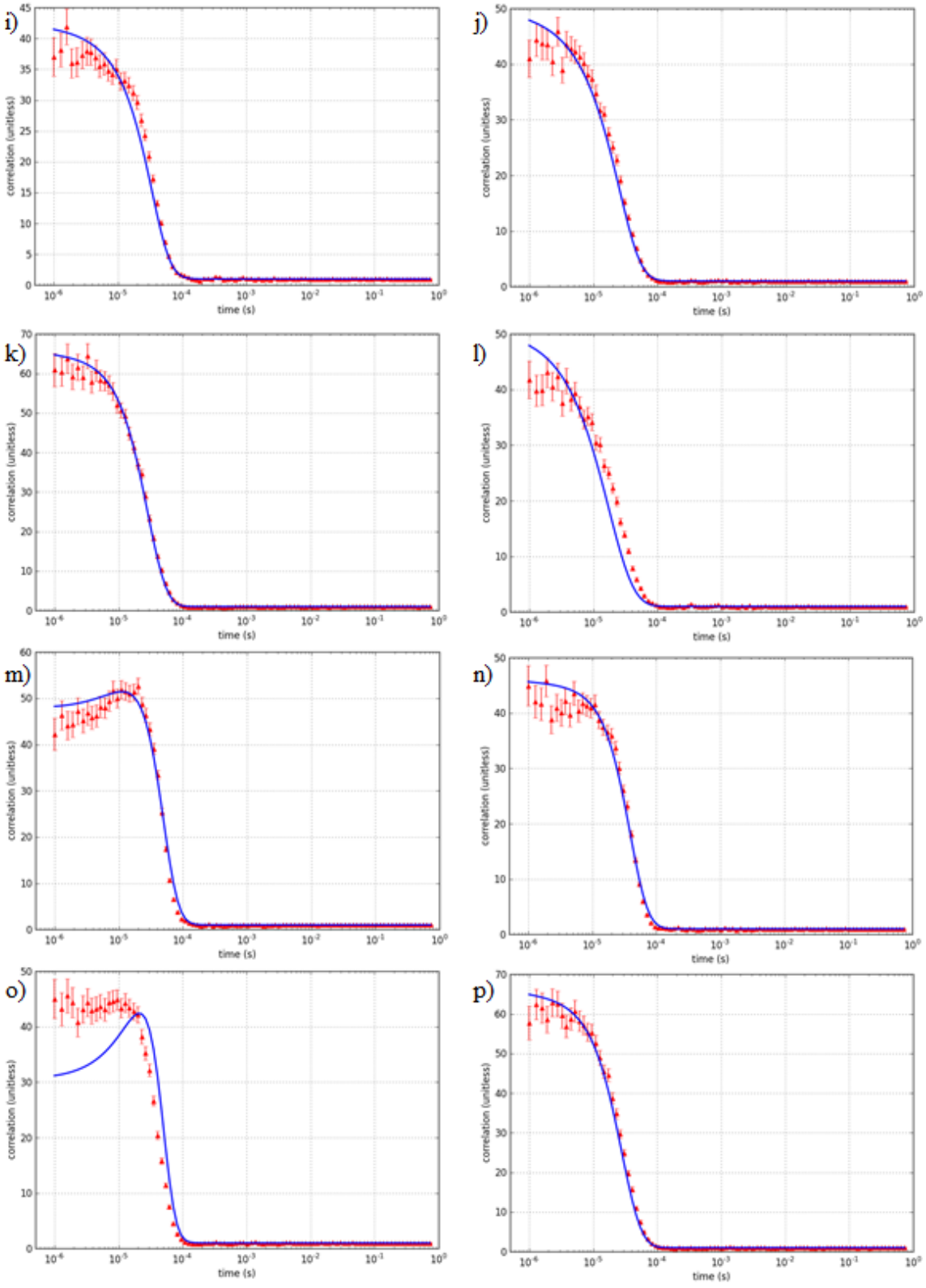


Figure 15. Continued


i)
j)
k)
l)
m)
n)
o)
p)
correlation (unitless)
time (s)

CHAPTER V-CORRELATION FITTING DISCUSSION

5.1 Gaussian-Lorentzian Model vs. 3D Gaussian Model

In the cases of both one focus and two foci, the Gaussian-Lorentzian model proved to be more accurate in determining the flow speed. This is not in itself surprising, because the Gaussian-Lorentzian profile more closely matches the actual laser intensity profile near the focus. The average difference between actual and fitted flow velocities for the 3D Gaussian model in the one-focus case was 9.9 mm/s while for the Gaussian-Lorentzian model it was approximately 0.7 mm/s, regardless of whether the triplet state parameters were included in the fitting.

In the case of two foci, the average difference for the 3D Gaussian model was -2.1 mm/s and for the Gaussian-Lorentzian model 0.3 mm/s. Although the Gaussian-Lorentzian model is more accurate, the amount of computation time required to fit the Gaussian-Lorentzian is substantial, taking ~1 hour to fit correlation functions (eq.2.20). This time increases considerably when more focal volumes are introduced to the model, with the number of correlation functions computed equal to the square of the number of focal volumes introduced.

5.2 Error for 3D Flow Velocity Measurements

As seen in Figure 13, both the 3D Gaussian and the Gaussian-Lorentzian models give results that approximately follow the simulated flow velocity, but there are noticeable, large discrepancies and a lack of consistent accuracy. One possible reason for these differences is that the simulated concentration was only 100 pM, as it had to be low due to limitations of the simulations, and this caused the number of photons in each data set to be only 50k–100k. The

low number of photons leads to large statistical errors. By contrast, the number of photons counted in experimental FCCS measurements is typically greater than 10^6.[9] Better results could possibly be obtained by combining data sets from many simulation runs to form correlation curves with better statistics.

Another possible reason for the lack of accuracy in the recovered flow velocity components is that the effects of cross-talk are too large or are not adequately accounted for. The choice of fluorescence lifetime, 1.8 ns to model Rhodamine B, may be too long in that it leads to a large amount of cross-talk. The correlation function that corresponds to two photons generated from foci *i* and *j* respectively ($i = j$ for the autocorrelation case) falling within the time bins associated with foci *i* and *j* only accounts for 71% of the fitted correlation function in the four-focus case, with the remaining 29% percent adjusted from crosstalk terms. Contrasted against this is the simulation for the case of only two foci, where the correctly counted correlation function accounts for 95% of the fitted function. It is also possible that the cross-talk is not adequately accounted for by eq. 3.1, as this does not account for the time response of the detector, which was set to be 0.3 ns full-width at half maximum in the simulation.

CHAPTER VI-BACKGROUND OF SINGLE-MOLECULE TRACKING

6.1 Single-Molecule Studies

Single-molecule studies are useful because these studies have the ability to detect states not normally available in ensemble measurements, they remove the need for synchronizing many molecules undergoing time-dependent processes, and they can probe new regimes for new processes and characteristics.[35] Experiments such as single-molecule Förster resonance energy transfer,[40] single-molecule polarization microscopy,[41] atomic force microscopy,[42] and single-molecule tracking have found use in determining these molecular characteristics. Studies involving biological molecules have especially benefitted from single-molecule studies,[43] and single-molecule tracking has been used to study the method of movement through cell membranes[44] as well as determine rates of diffusion.[45] Methods of tracking can be grouped according to the fluorescence detection method: wide-field and confocal.

6.2 Wide-Field Tracking

Wide-field imaging illuminates a large portion of a sample. In turn, multiple fluorescent particles may be excited and fluoresce. The illumination can be provided either by forward illuminating a sample or by Köhler epi-illumination. The resulting image is detected with a camera, where Charge-Coupled Device (CCD) cameras are often used. The main advantage to using a wide-field microscope in imaging a sample is that particles can be resolved to a very high precision [46-49] in the focal plane. The techniques of resolving and localizing particles vary,[46, 48] among them being Gaussian fitting,[49] image cross-correlation,[48] and center-of-mass calculation.[48] Multiple particles can be imaged with the wide-field method and, with

simultaneous imaging of multiple focal planes, particles can be tracked as they move in three dimensions.[50]

Normally, the point spread function (PSF) of a fluorescent particle, estimated as a point emitter, is an Airy function[46] in-plane. The size of the PSF increases as the particle is displaced from the focal plane, but the increase is symmetrical for displacement in either direction. Because of this symmetry, it is not normally possible to unambiguously track the axial motion of a particle as it moves out of the focal plane.[46] Fortunately there are workarounds to this particular problem. The shape of the PSF may be modified so that it changes asymmetrically as the emitter moves on either side of the focal plane. This may be accomplished by using astigmatic imaging [51] or spatial light modulation of the PSF into a double helix.[52]

There are disadvantages to wide-field imaging. In thick (~100 μm)[53] samples, the ratio of fluorescence signal to background is lower in wide-field microscopy than in microscopy techniques that section the sample axially, e.g., confocal microscopy[53] and light sheet microscopy.[47] Background fluorescence generated outside the desired focal plane, as well as from the biomolecules naturally found in the sample,[54] generate degradation of collected images.[47] Higher rates of phototoxicity in the sample are also a concern, [55] more so in wide-field microscopy because the entire sample is illuminated, as opposed to techniques that illuminate a smaller sample section.

6.3 Confocal Tracking

In contrast to wide-field microscopy, confocal microscopy uses a focused laser beam to excite particles, which results in a depth discrimination mechanism. This depth discrimination from a tightly focused excitation beam exists for both one-photon and two-photon excitation, but

it is accentuated for two-photon excitation. In either case, the depth discrimination can be improved by using a pinhole in the image plane, with size and position confocal to the focused excitation.[54] The pinhole is typically not used for two-photon excitation, as the natural depth selection of the focused beam with multi-photon absorption is usually sufficient.[7] For one-photon excitation, the pinhole can similarly be omitted or increased in size, but this usually results in a decrease in depth resolution.

The detectors used in confocal microscopy, typically small-area avalanche diodes, have a high temporal resolution when compared to CCD cameras (sub-ns[32, 56] for diodes as compared to ms [54] [full frame] and μs[57] [partial frame] for cameras). As stated before, the signal-to-noise ratio from a confocal microscope is generally better in thick samples than that from wide-field microscopy, especially in cases where photobleaching is a concern.[58] The general drawback to confocal microscopy, especially in terms of localizing a particle, is that the resolution of a single focused beam is limited to the Abbe diffraction limit.[59] Recently, there has been a push to beat the diffraction limit with the use of specialized metamaterial lenses,[60] but imaging below the diffraction limit can also be achieved with variations on the confocal microscope. These include methods in which the positions of single emitters are measured to a precision below the diffraction resolution limit.

One manner of sub-diffraction localization uses confocal scanning microscopy. A focused laser beam is swept across a sample (or conversely, a sample is swept through the beam) where molecules of interest are tethered to a glass surface,[61] held statically in vivo,[62] or slowly diffuse in gels.[63] The location of each emitter is then found from the center of its PSF. This method provides submicron resolution in the case of little to no molecular movement where

the PSF can be accurately measured.[64] Another method of sub-diffraction localization applicable to moving emitters uses the orbiting spot technique.[26, 65, 66] A focused laser beam orbits a point in space, which is accomplished by employing galvo-motor driven scanning mirrors or a crossed pair of acousto-optic beam deflectors and comparing the signal generated from different parts of the orbit.[65] This method can be extended to achieve localization in 3D by alternating the orbit between two planes along the optical axis and measuring the relative signal between them.[66] This has been accomplished by mounting the microscope objective on a piezoelectric stage, but the period of alternating is then quite slow. An alternate method uses a single circular orbit but splits the collected fluorescence into two detection arms, which use pinholes set to image different focal depths.[43] Localization accuracy can be adjusted by varying the orbit radius.[65] Another method of localization is to establish a 3D geometry of multiple detection volumes via four fibers, whose cores serve as confocal pinholes, connected to four photon detectors.[67-69] This four-pinhole technique is capable of tracking particles with a diffusion constant up to 1 $\mu m^2/s$. [68] Trajectories of quantum dots, both in solution[67] and on a cell membrane,[69] have been demonstrated. Quantum dot tracking in a high-background (low signal-to-noise) solution has also been shown.[33]

The research presented in this chapter details an alternative configuration to the four-pinhole technique. Instead of defining the observation areas with physical apertures, the sample space can be defined with four alternating excitation volumes. All fluorescence is collected with common SPADs and is sorted according to which excitation volume generates the detected fluorescence. By comparing the relative fluorescence response from each focus, the position of a fluorescent particle can be localized to below the diffraction limit quickly and in real time. In

principle only one detector is needed, but in our set-up, two detectors that respond to different polarizations are used and their counts are combined. Because the localization method does not require the collected fluorescence to be split at a beam splitter to separately image two different focal planes, it should be capable of a higher overall efficiency of detection than the four pinhole method[68] or the circular orbit with two image plane method.[43] Therefore, in order to document this advantage, FCS measurements of the overall detection efficiency are reported in section 8.2.

CHAPTER VII-MOLECULE TRACKING APPARATUS

7.1 One-Photon Setup – Four Alternating Laser Diodes

Four 635 nm laser diodes (Lasermate model No. T63E-P-FC121-S4) were used as the illumination sources for the one-photon setup. The spectral output of the four diodes, measured with a compact spectrometer (Ocean Optics, model HR4000CG-UV-NIR), is shown in Figure 16. Whereas the beam from a laser diode is typically asymmetric and multi-mode, fiber-coupled diodes were chosen for this experiment because the fibers spatially filter the beams to enable them to be tightly focused to diffraction-limited spots. While the inputs of the fibers were directly attached to the diodes, the fiber outputs were SMA connectors, which were screwed onto fixed-focus fiber collimators (Thorlabs, a discontinued model similar to model F220SMA-B). Each diode was soldered to an automatic power control (APC) board (Lasermate model No. APCT-42X), which allowed for TTL modulation and also had a trim pot for adjusting current to the diode. Wires for the TTL input signal and power supply circuit were soldered onto the APC control boards. The four diode and APC board sets were fixed to a piece of perfboard. Each positive lead of the APC boards was soldered to a 56.2 Ω resistor and each TTL control wire was soldered to a 5.111 kΩ resistor. A common power supply (BK Precision 1787B) provided a 5 V bias to the diodes with 50 mA maximum current. The four TTL control wires were soldered into a 68-pin connector, which was plugged into a National Instruments PCI-6602 counter/timer card. A wire was connected from the power supply to the NI card's ground so that the counter/timer card and the power supply shared a common voltage reference. A simple circuit diagram of the diode electrical setup is shown in Figure 17, and photos of the diode circuit board are presented in Figure 18 and Figure 19.

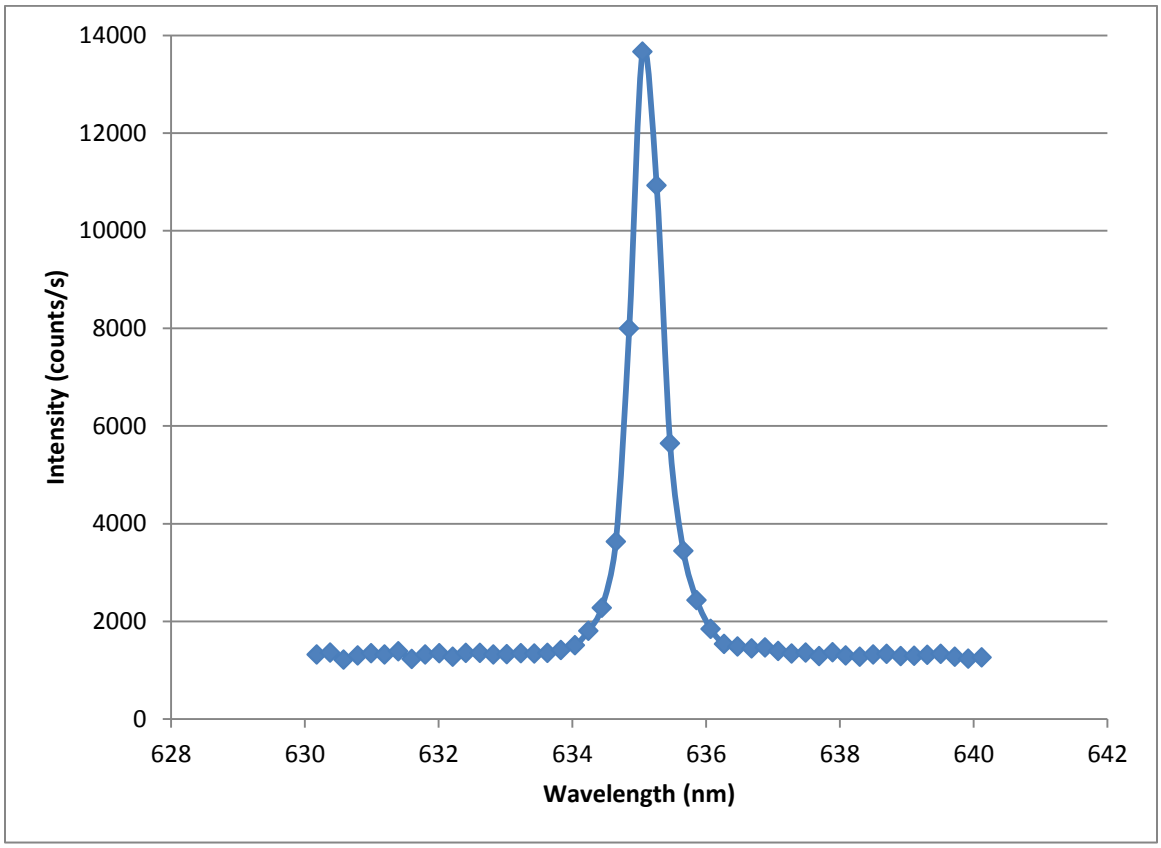


Figure 16. Spectral output of the diodes. Peak output is centered at 635 nm.

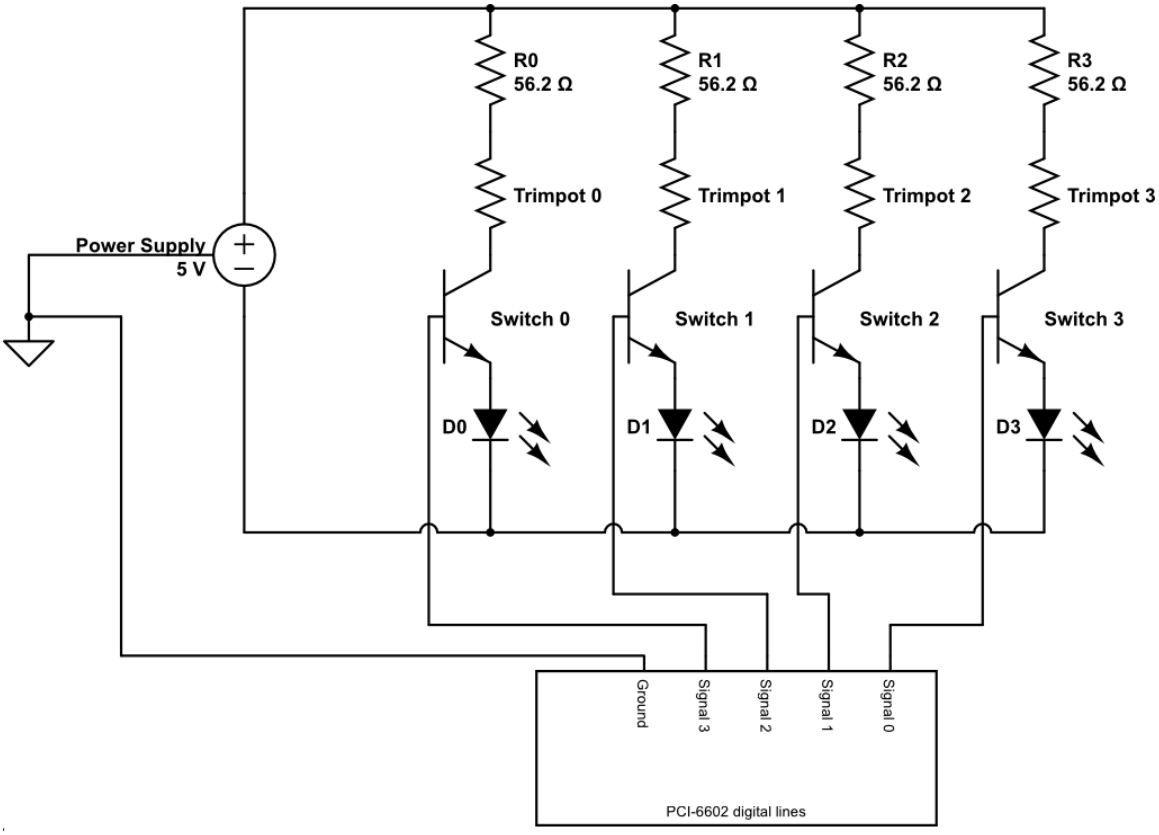


Figure 17. Circuit diagram for the laser diode system. The power supply provides a constant 5 V potential with a maximum current of 50 mA. Individual currents are adjusted with the trim pots. Typically, the three diodes that are off draw a combined 8.4 mA of current. The current from the control wires connected to the PCI-6602 card was measured to be <100 µA. A common ground was shared between the PCI card and the power supply to ensure the switch signal was at 5 V.

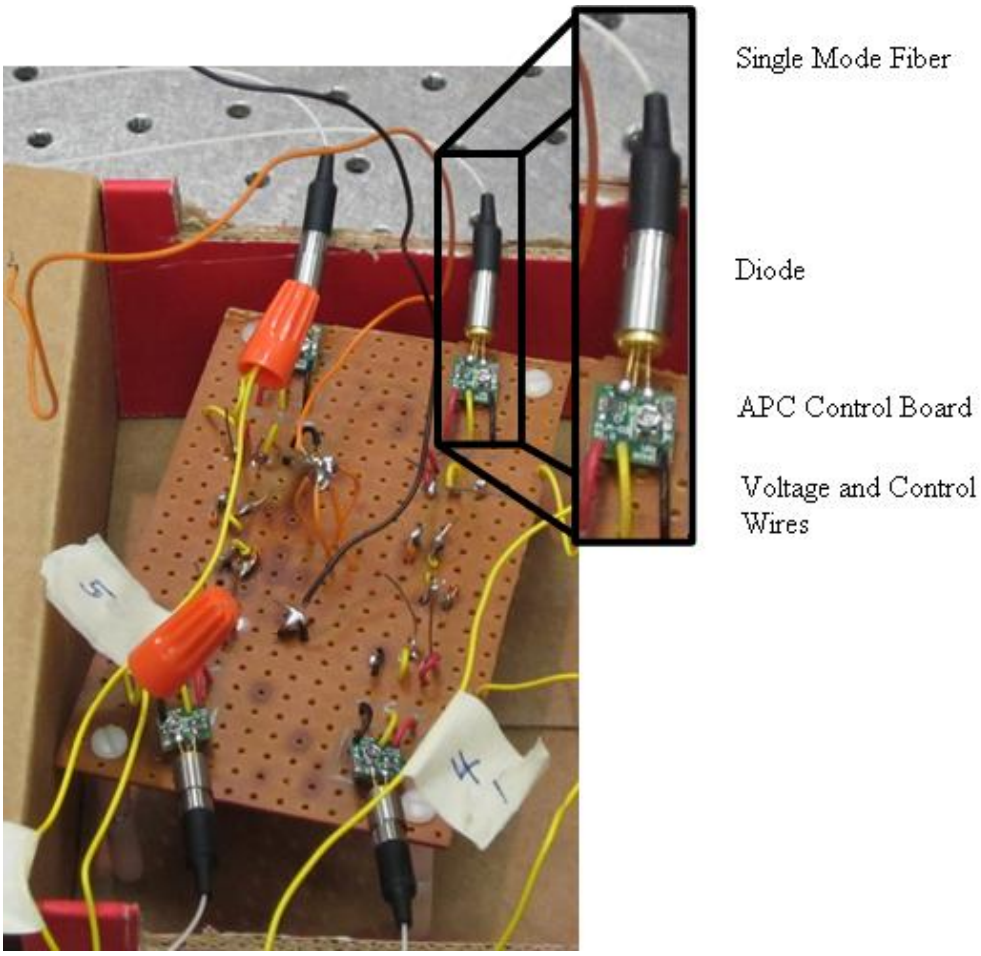


Figure 18. Front of diode circuit board. The four diodes are soldered into the four corners of the perfboard. The red wire on the APC board is the positive lead, the black wire is the negative lead, and the yellow wire is the control wire. The leads (positive and negative) from all four APCs are soldered together (center of radio board). The wires from the power supply and the PCI-6602 card are connected with wire screws (orange caps) for ease of assembly/disassembly.

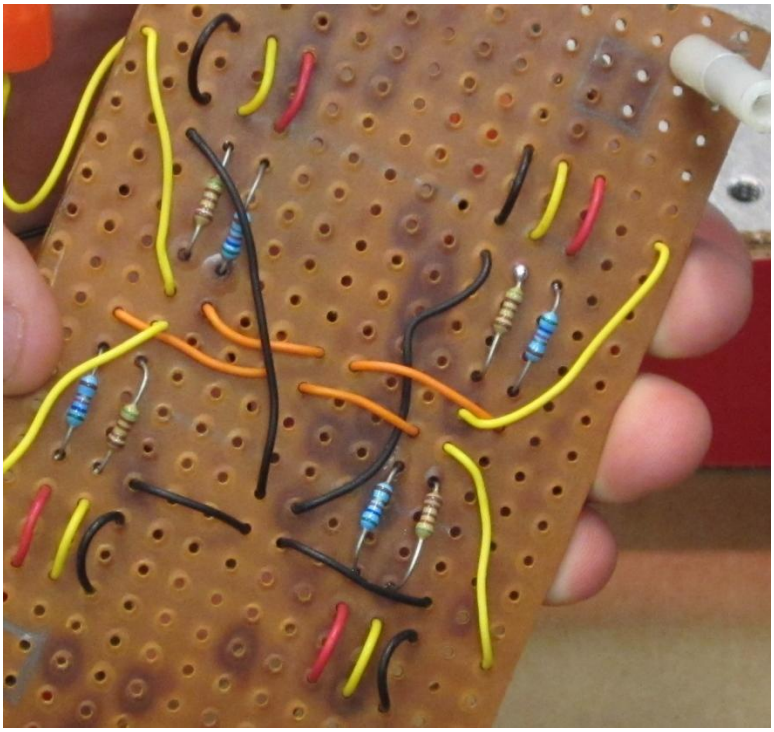


Figure 19. Back of the diode circuit board.

7.2 One-Photon Optics

The four collimators were screwed into holders on the optical table, as shown in Figure 20 a) and b), spacers were inserted into two of the four collimators. After each collimator, the beams were slightly converging and came to a focus at distances of 0.85 m, or 3.12 m for the case of the collimators with, or without the spacers respectively. The four beams were combined with the use of three 50/50 beam splitters (Thor Labs BSW10) to produce four almost parallel and almost collinear, overlapping beams. The combined beams were passed through a linear polarizer (Oriel PHLL-10) to ensure that each laser diode would give the same polarization at the sample. Figure 20 a) shows the beam splitter configuration.

At a distance of 1.33 m from the collimators, the beams have the same size, but those from the collimators with spacers are diverging while those from the collimators without spacers are converging. This point is imaged to the entrance pupil of the objective with a 3× magnification by the use of two suitably placed lenses, a 100 mm focal length lens (Thor Labs A0254-100-A1) and a 300 mm focal length lens (Thorlabs LA1484, or similar). The placing of the lenses was determined using Zemax optical design software. This configuration achieves excitation at points that are on either side of the plane that is confocal to the pinhole in the detection arm of the microscope.

Before the beams enter the microscope, the height of the beams above the optical table is raised with a periscope, which also rotates the plan of polarization. The beams then reflect from the uncoated surface of a fused silica substrate (Newport 10B20-01NC.2) towards the objective (Olympus UPLSAPO 60×, N.A. = 1.2, water immersion), which is positioned in an inverted configuration. The Fresnel reflection coefficient from the substrate surface for the p-polarization

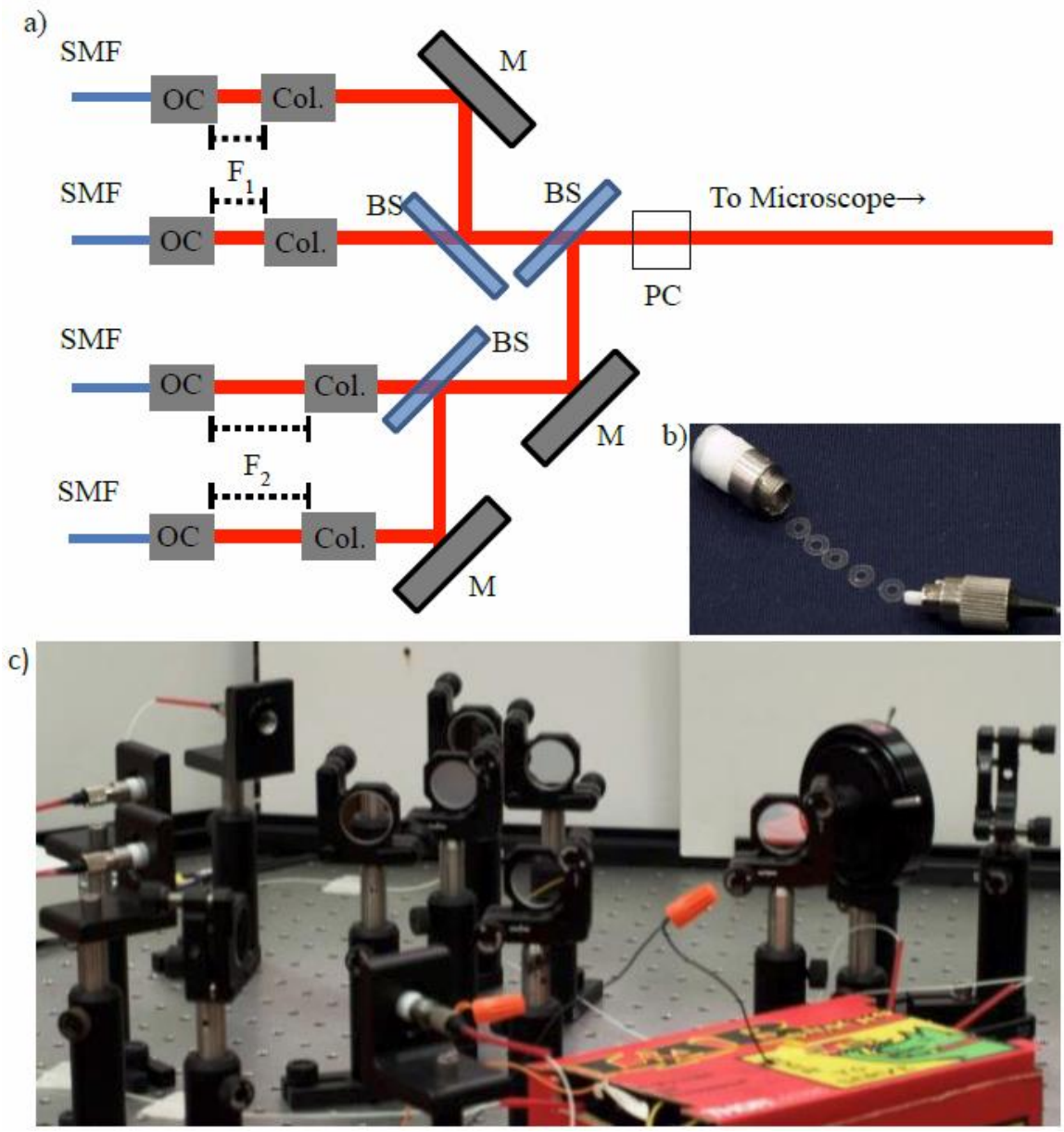


Figure 20. a) Sketch of the beamline combination optics. The diode light passes from the single-mode fibers (SMF) into output couplers (OC). The beams are then focused with collimators (Col.). Two of the collimators are placed at a distance F1 and the other two are at F2 from the OCs. The beams are either reflected by or transmitted through three 50/50 beam splitters (BS) with the aid of steering mirrors (M). The combined beams pass through a linear polarizing cube (PC) and proceed to the microscope. b) Photograph of (from left to right): beam collimator, 5 100 µm spacers, and output coupler. c) Photograph of the experimental beamline combination optical setup.

is close to 10%, and the remaining 90% of the power from each beam passes to a beam dump.

7.3 Fluorescence Collection

Fluorescence is collected with the same objective and hits the fused silica substrate. The Fresnel reflection coefficient for s-polarization is close to zero, so the net throughput for unpolarized fluorescence is close to 95%. The collimated beam of fluorescence then passes through an interference filter (Omega 3RD660-740) to isolate the fluorescence from scattered 635 nm excitation light and Raman-scattered light from the solvent. The beam is then focused with a 250 mm focal length plano-convex lens onto a pinhole, which results in an overall magnification of 83.3×. For initial setup, the beam may instead be focused onto an electron multiplying (EM)-CCD camera.

The pinhole is 150 μm microns in diameter, which is larger than the imaged diameter of the excitation volume from each beam (83.3 μm). It serves to improve signal to background by blocking scattered light originating far from the beam foci, but it does not provide significant depth discrimination of collection from each of the four beams. Depth discrimination arises mostly from the tightness of focus of each beam.

A 150 mm focal length lens located after the pinhole recollimates the light, which then passes through a polarizing beam splitter. Each polarization arm contains an 8 mm focal length asphere (Thorlabs C240TME-B) that focuses the fluorescence onto a SPAD detector. Both detection arms are of identical length. Transistor-transistor logic (TTL) pulses are sent from the SPADs through 50 Ω cables to the same PCI-6602 card which controls the laser diodes. The collection apparatus is shown in Figure 21.

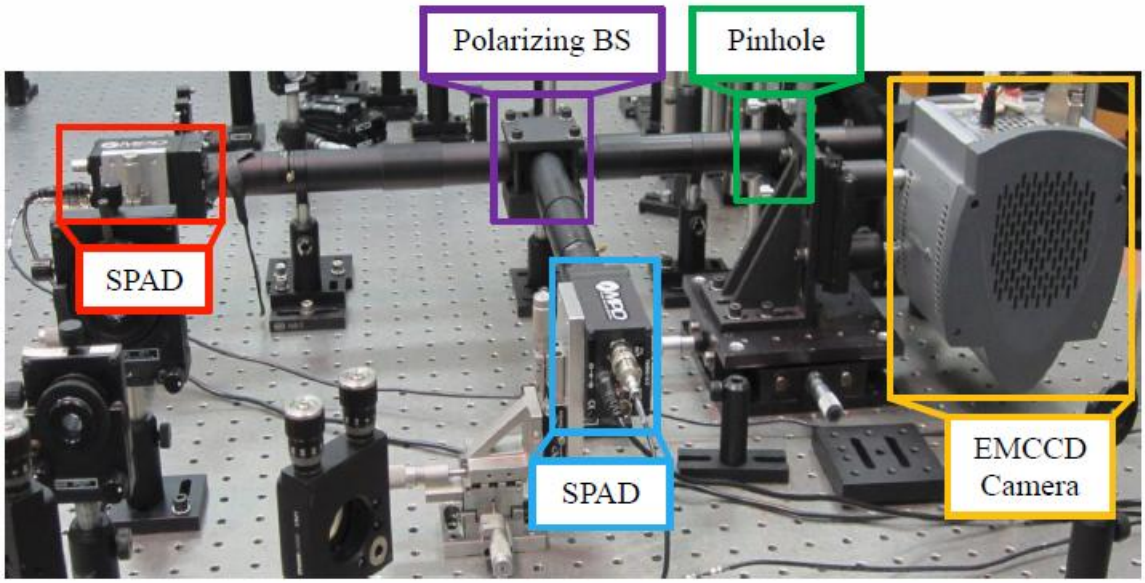


Figure 21. Fluorescence collection arm. Fluorescence generated at the sample is focused either onto an EMCCD camera for focal position adjustment or through a pinhole and onto SPADs.

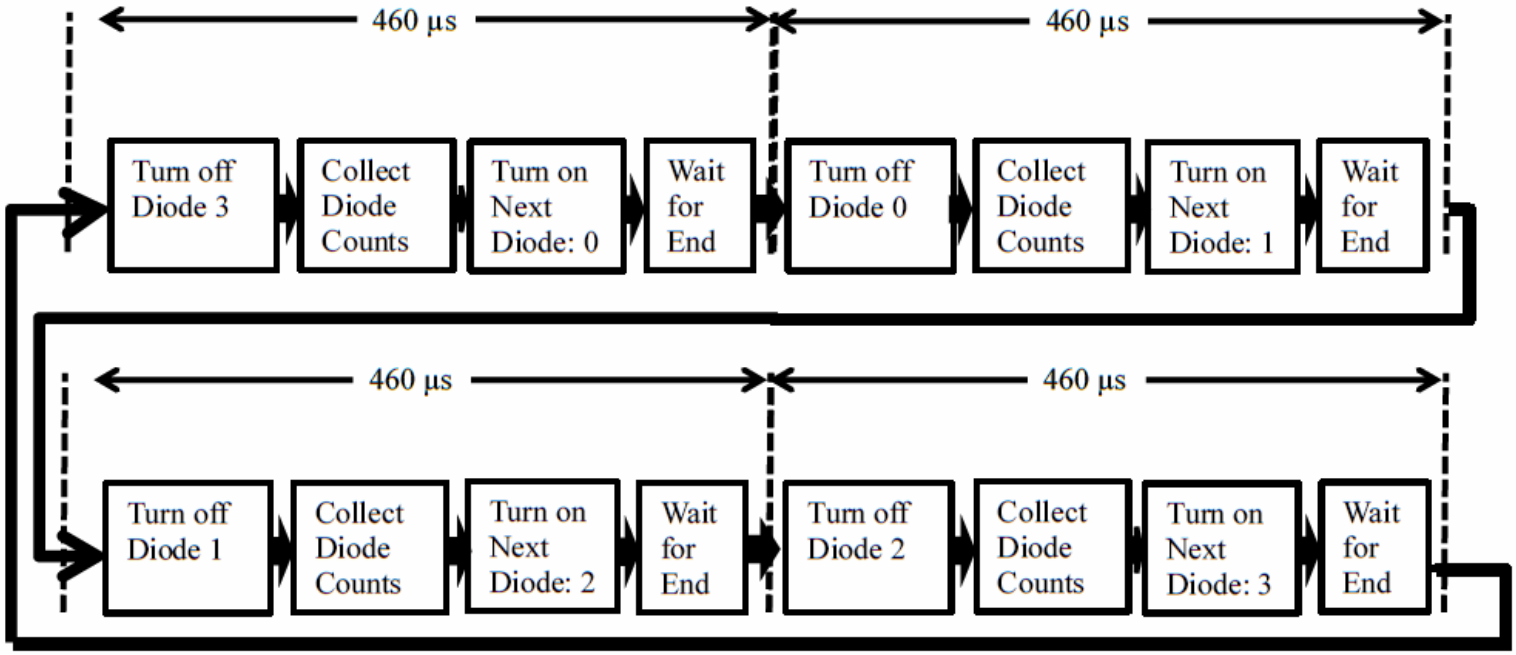
7.4 Feedback System

The feedback system is controlled by a LabVIEW program that executes under the LabVIEW Real-Time operating system on a so-called Target PC. This program has three loops. The first loop modulates the diodes and counts TTL pulses sent from the SPADs into four time-gated channels, in synchrony with the modulation of each of the four diodes. The second loop, which is synchronized to the first, uses the photon counts to estimate the position of a fluorescent nanoparticle with respect to the center of the four foci and it sets the digital output lines to command the piezo stage (Physik Instrumente P-733.3DD) to move in the counter direction to return the nanoparticle towards the center. The third loop, which runs at lower priority, sends the photon counts from each of the four beams and the piezo movement data to the so-called Host PC, over the network. It also accepts changes in operating parameters sent over the network from the Host PC. The three loops of the Target machine—the PCI-6602 loop, the piezo loop, and the data transmission loop—are described in more detail in the following sections.

7.4.1 PCI-6602 Loop

This loop uses six digital input/output lines from the National Instruments PCI-6602 card. Four of these lines, designated as lines 8, 12, 16, and 20, provide the "on" signal for modulation of diodes 0, 1, 2, and 3, respectively. Two of the lines, designated as PFI 38 and PFI 35, are the inputs of counters 0 and 1, respectively. The counter lines are set to count received TTL pulses from the SPADs. The loop, as shown in Figure 22, has four frames. The execution time between the start of frames is set to be 460 μs, which is one quarter of the empirically determined minimum time between commands to the piezo stage. The order of execution in

Figure 22. Flow diagram of PCI-6602 loop. At any time only one diode is on. The basic process of the cycle is that the previous diode turns off, counts are collected, and the next diode is turned on. Before the next cycle starts, the program waits for the 460 µs iteration time to finish.


460 µs
460 µs
Turn off Diode 3
Collect Diode Counts
Turn on Next Diode: 0
Wait for End
Turn off Diode 0
Collect Diode Counts
Turn on Next Diode: 1
Wait for End
460 µs
460 µs
Turn off Diode 1
Collect Diode Counts
Turn on Next Diode: 2
Wait for End
Turn off Diode 2
Collect Diode Counts
Turn on Next Diode: 3
Wait for End

each frame is as follows: the diode turned on in the previous frame is turned off; the number of TTL pulses recorded from SPADs 0 and 1 are written to a shared variable of the diode just turned off; the next diode is turned on; and the loop waits for the remaining frame time before executing the next frame. The counts from SPAD 0 and SPAD 1 are combined into a single 32-bit variable to ensure simultaneous transfer between loops and between computers.

7.4.2 Piezo Loop

This loop takes the counts binned for each diode, computes the relative particle position, and sends commands to a piezoelectric stage. The count rates are first read out and compared to a threshold limit, set in these experiments at 10 counts/1.84 ms. When the number of counts is less than the threshold, the piezoelectric stage raster scans the sample, equivalent to sweeping the four foci through the sample until a fluorescent nanoparticle is found and the threshold limit is exceeded. The total scan range of the piezo is 30 µm × 30 µm × 10 µm. The limits for the raster scan to find a particle are user adjustable and are typically set at 20 µm × 20 µm in-plane (*x*, *y*) and 4 µm on-axis (*z*).

The summed count rates, when greater than the threshold, are used in a linear approximation calculation for estimating the particle position with respect to the center of the four foci. The position estimates are calculated from

$$x' = \frac{N_0\Delta x_0 + N_1\Delta x_1 + N_2\Delta x_2 + N_3\Delta x_3}{N_{tot}}, \quad 7.1a$$

$$y' = \frac{N_0\Delta y_0 + N_1\Delta y_1 + N_2\Delta y_2 + N_3\Delta y_3}{N_{tot}}, \quad 7.1b$$

$$z' = \frac{N_0\Delta z_0 + N_1\Delta z_1 + N_2\Delta z_2 + N_3\Delta z_3}{N_{tot}}, \quad 7.1c$$

where N_i is the number of counts collected when diode *i* is on, N_{tot} is the combined number of photon counts from diodes 0, 1, 2, and 3, and Δx_i, Δy_i, and Δz_i are the distances from the center of the tetrahedron to focus *i*:

$$\Delta x_i = x_i - x, \quad 7.2a$$
$$\Delta y_i = y_i - y, \quad 7.2b$$
$$\Delta z_i = z_i - z, \quad 7.2c$$

where

$$x = (x_0 + x_1 + x_2 + x_3)/4, \quad 7.3a$$
$$y = (y_0 + y_1 + y_2 + y_3)/4, \quad 7.3b$$
$$z = (z_0 + z_1 + z_2 + z_3)/4, \quad 7.3c$$

are the midpoints along the respective axes. The coordinates of the position estimate are multiplied by user-adjustable proportionality constanst K_x, K_y, K_z, so that the piezo is commanded to move a step size of

$$x_{step} = -K_x x', \quad 7.4a$$
$$y_{step} = -K_y y', \quad 7.4b$$
$$z_{step} = -K_z z'. \quad 7.4c$$

When a new position is sent to the piezoelectric stage, the stage begins moving toward the new position, but it is possible that the piezo stage does not complete its move before the next movement command. When the photon counts are above the threshold, a flag is turned on to indicate that the tracking routine is now in effect, as opposed to the raster scan routine.

A tracking process can end in three different ways: the particle is lost; a physical travel limit of the piezoelectric stage is reached; or the tracking program is terminated. The particle is lost when the total number of photons per 1.84 ms cycle drops below the set threshold. After initially losing the particle, the piezoelectric stage remains stationary for up to seven cycles. If the threshold is exceeded again during the seven cycles, the tracking routine resumes.

Otherwise, a new raster scan search commences until the threshold is once again exceeded. In the case where the stage reaches the limit of travel along one or more axes, the tracking routine will continue moving the piezoelectric stage until the next step would move the stage beyond the physical movement range. The piezoelectric stage then moves to the middle of the range of motion for all three axes and begins a new raster scan to find a new particle. The tracking program is set to run for a 3.5 minute time block. This programmatic cutoff time, which may truncate some tracking events, was chosen in the interest of preventing overflow errors in data collection/recording. The entire process is show in Figure 23.

7.4.3 Data Transmission Loop

Data is transferred from the piezo loop to the data transmission loop on the Target PC using a Real-Time FIFO buffer, and then from the data transfer loop to the Host PC using a network data stream. Only when processing resources are available is data transferred, so as to not interrupt collecting photon counts or commanding the piezoelectric stage movement. On the host computer the data transferred over the network is written to a binary file, which may be read by another program for post analysis or writing to a spreadsheet.

7.5 Spreadsheet Reader and Post-Processing

A Python script (Appendix B) was coded to interpret the data written to the spreadsheet. The flags were checked for the beginnings and the ends of tracking runs. When the flag, which turns "ON" at the beginning of a tracking run changes to "OFF," the script checks to see if the particle was lost, if the particle passed beyond the limit of the piezoelectric stage travel, or if the tracking run ended programmatically. In the case of the particle being lost, the script checks the

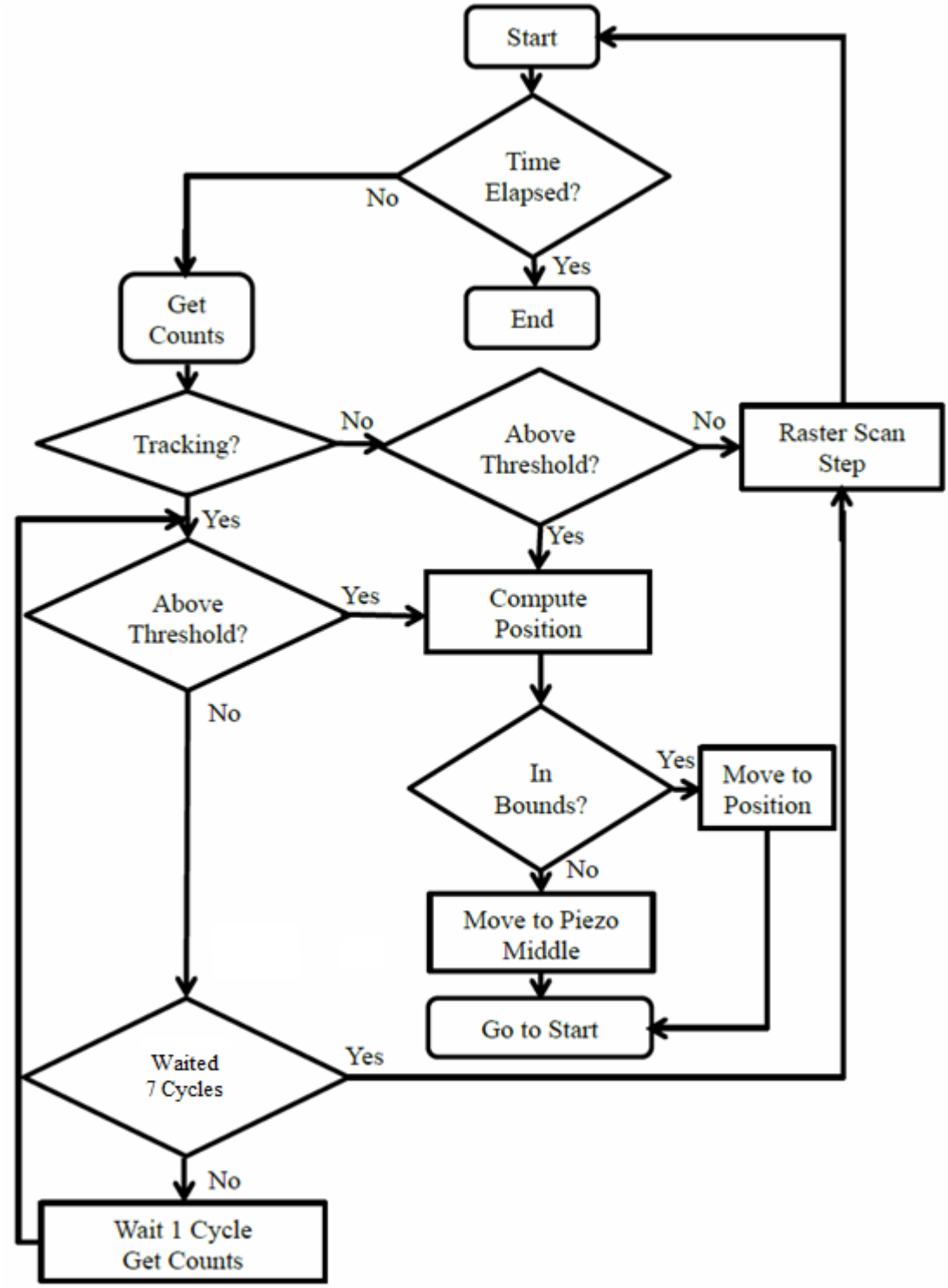


Figure 23. Flow chart of piezo loop process. The decision on tracking braches on whether the previous iteration was tracking a particle or not.

following ten rows to determine if the flag turns back ON. If the flag does turn back ON, the positions of the OFF flag and ON flag are compared to see if their separation falls within the average mean distance the particle could move in the allotted time. For example, if the particle has a diffusion coefficient of 1 $\mu m^2/s$ and the time between the OFF and ON flags is 6 ms, from the 3D diffusion equation,

$$r^2 = 6D\tau, \qquad 7.5$$

if the distance traveled is less than 0.190 μm, then the two sets of data are assumed to be part of one larger set for the same particle.

7.6 Throughput Measurement

As discussed at the end of section 6.3, the four-focus method of tracking has the potential for achieving significantly greater optical collection and throughput of fluorescence than some alternate techniques that involve splitting the fluorescence to image different focal depths or positions. Therefore, in this section, the efficiency of fluorescence collection and detection is measured. The fluorescence collection efficiency could give a standard comparison between confocal single molecule tracking setups, however the collection efficiency of the previously published techniques, the four-pinhole and the revolving focus, are not reported in the literature to our knowledge. With the four-pinhole method, fluorescence light generated at the point conjugate to the center of the pinhole tetrahedral pattern will be blocked, detracting from the fluorescence efficiency. The two-plane revolving focus technique could in principle have high collection efficiency, but has a slower time for completion of a cycle and update of the 3D position measurement (32 ms).[66] Typically, if the apparatus is single-molecule sensitive, the collection efficiency is between 1%–8%.[35]

Fluorescence correlation spectroscopy measurements are performed to determine the throughput. A concentrated solution of streptavidin-conjugated Alexa 647 is placed on a coverslip and all four diodes are used to generate fluorescence. The diodes are modulated as they would be when tracking a particle. The autocorrelation function is measured and fit to determine the mean number of molecules within the excitation volume from the peak of the autocorrelation function. Also, the total count rate is measured to then find the mean rate of photons per molecule. The measured molecular fluorescence rate is then compared to the theoretical maximum possible molecular fluorescence rate to determine the net efficiency or throughput. Results are given in section 8.2.

CHAPTER VIII-TRACKING DATA

8.1 Raster Scan Data

Initial alignment was achieved by adjusting the beam splitters and mirrors in the laser diode assembly to evenly space the foci, as determined by using the EM-CCD camera to image the autofluorescence from a coverslip placed at the focal plane. Then a raster scan was performed to determine the focal positions in a sample. A fluorescently labeled latex bead embedded in polyvinyl alcohol (PVA) was moved through the focal volumes with step sizes of 80 nm perpendicular to the optical axis and 100 nm along the optical axis. The distances from the midpoint of the tetrahedron to the center of each focus are recorded in Table 7. The midpoint of each focus was determined as the point of greatest intensity. A map of the focal volumes is shown in Figure 24.

Table 7. Distances in nm from the midpoint of the tetrahedral excitation region to the respective foci. The axial distances have been adjusted to account for refractive index differences between PVA (n = 1.52) and water (n = 1.33).

Focus	Δx	Δy	Δz
0	0	250	-1200
1	0	-250	-743
2	-250	0	972
3	250	0	972

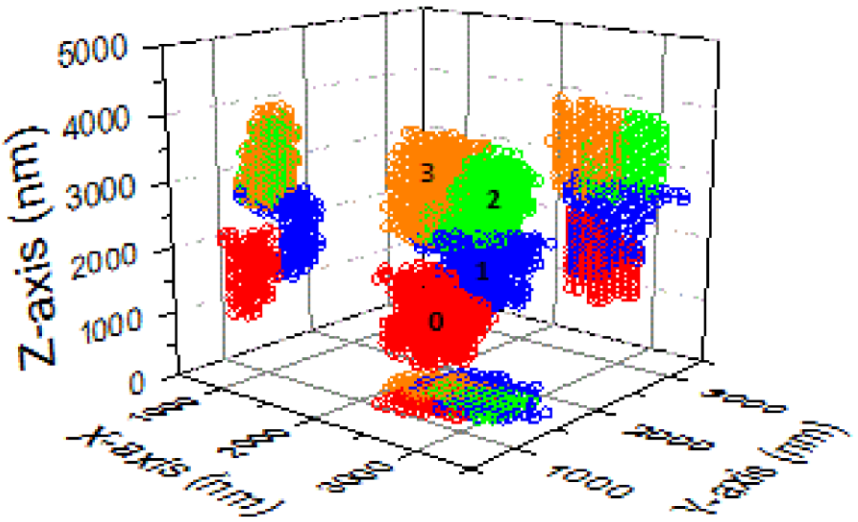


Figure 24. 3D map of the focal positions. Each symbol marks where the intensity of the focus is within 67% of the maximum intensity. The foci have been normalized to have equal maximum intensities for this map. The axis setup on the map matches the axis setup for the piezoelectric stage.

8.2 Throughput Data

Three FCS measurements were taken of the Alexa 647 conjugates freely diffusing for 30s. The correlation functions were averaged and the amplitude of the peak in the autocorrelation function was determined by fitting it to a 3D-Gaussian model with triplet component, as seen in Figure 25. The total number of photons accumulated in the 90 s collection time in the autocorrelation data was 4,563,668, corresponding to a mean count rate (F) of 50,700 /s. The background count rate was separately measured to be 186.01 ± 2.41 counts/s. If the background count rate is much lower than the fluorescence count rate, as is the case here, the amplitude of the autocorrelation $g(0)$ is one plus a factor proportional to the reciprocal of the mean number of molecules in the excitation volume, N, i.e.,

$$g\ 0\ = 1 + \frac{1}{2^{1.5}N} \qquad 8.1$$

The autocorrelation peak value from the fit was $g(0)$ = 1.66 ± 0.03, which gives N = 0.54. The mean fluorescence count rate per molecule (R_M) is thus

$$R_M = \frac{F}{N} = \frac{50{,}500}{0.54} = 94{,}300\ \mathrm{s}^{-1}\mathrm{molecule}^{-1}. \qquad 8.2$$

The throughput of the microscope can be determined by comparing the above value with the calculated rate of fluorescence per molecule, assuming that there is 100% efficiency. The optical powers of the four diodes before reflecting off the fused silica substrate are noted in Table 8. The combined power of the laser diodes was 140 μW, making the power incident in a sample of 5 nM Alexa 647 as P = 14.0 μW. This sets the rate of incident photons into the sample at 4.5×10^{13} photons/s. The average excitation rate (R_E) for a single molecule in the focal plane of an excitation beam is

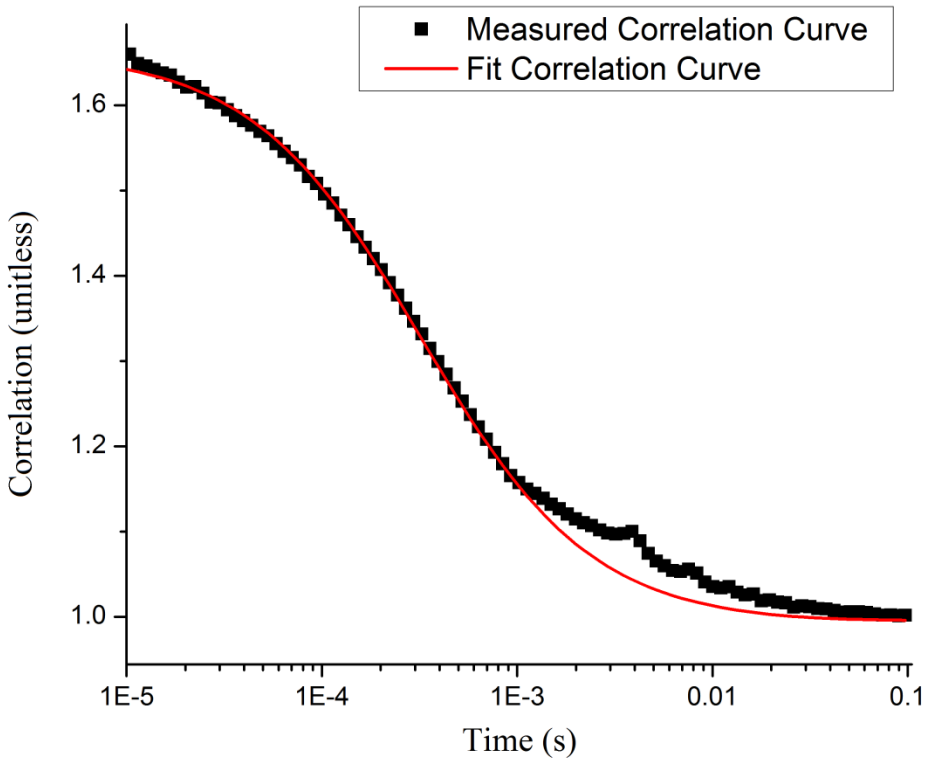


Figure 25. Averaged correlation function and fit. The fitted peak has a value of 1.66 ± 0.03. Discrepancies between the correlation curve and the fit arise from triplet state excitations (1–100 μs) and from correlation artifacts arising from diode modulation.

Table 8. Diode laser powers. The powers were measured before the beam expander and fused silica substrate on the optical table.

Diode	Power (μW)
0	34
1	36
2	35
3	35

$$R_E = \frac{2P\sigma}{E_\gamma \pi w_o^2}, \tag{8.3}$$

where E_γ is the photon energy, sigma is the absorption cross section, and w_o is the beam waist. In these experiments E_γ = 3.12 × 10^{-19} J for a photon at 635 nm; σ = 0.458 × 1.03 × 10^{-15} cm^2, where the first factor accounts for the decrease in absorption between the peak at 650 nm and the excitation wavelength of 635 nm, while the second factor is the absorption cross-section of Alexa 647 at peak excitation of 650 nm; and the beam waist determined from the fit of the autocorrelation function is w_o = 447 nm.

The average rate of fluorescence from a molecule in the focal plane of an excitation beam would be the above excitation rate times the fluorescence quantum yield, which is Φ = 0.33 for Alexa 647. This gives a value of

$$R_f = \Phi * R_E = 2.23 \text{ x } 10^6 \text{ photons s}^{-1}. \tag{8.4}$$

Note that the rate would be lower if the molecule were displaced from the focal plane, as the size of the beam becomes greater than w_o. A lower limit for the net throughput *T* or efficiency of collection and detection is thus given by

$$T = \frac{R_M}{R_f} = 0.042. \tag{8.5}$$

This value is well within the 1-8% efficiency range of typical single-molecule microscopes.[35]

8.3 Latex Bead Tracking

Latex beads were suspended in six different solutions; 100% glycerol; aqueous solutions containing 72% glycerol by mass, 63% glycerol by mass, 50% glycerol by mass, and 37% glycerol by mass; and pure distilled water. In all solutions the concentration was 362 fM. The diffusion constants in each solution are listed in Table 9, as derived by Cheng (2008).[70] Also given in Table 9 is the mean time that a bead would take to diffuse from the center of the four foci to beyond the tetrahedron. The diffusional time is considered only in 2D, i.e,

$$\tau = \frac{r^2}{4D}, \qquad 8.5$$

where *D* is the diffusion coefficient and *r* is the distance on plane from the center of the tetrahedron to the outer edge of the foci. The distance *r* was taken to be 697 nm (250 nm from the center of the tetrahedron to the center of a focus and 447 nm from the center of the focus to the edge of the tetrahedron). 2D diffusion was chosen because the lateral distance (647 nm) was almost an order of magnitude smaller than the distance from the center of the tetrahedron to the edge of the tetrahedron along the optical axis (~4 μm).

In analyzing the tracking data, if a tracking time was less than 18.4 ms (10 cycles), then the tracking run was not counted toward analysis as it was most likely caused by a fluctuation in the background or a fluorescent particle passing though the periphery of one of the beams. Average tracking times using the four-focus setup and piezoelectric stage are noted in Table 9. Average tracking times were at worst three times longer than the mean time a bead would take to diffuse out of the tetrahedral region. The average factor of increase in occupation time was ~5.

Histograms of tracking times are shown in Figure 26 and Figure 27. Additionally, examples of tracking runs are shown in Figure 28-Figure 33 with position measurements and photon counts at each cycle of time.

Table 9. Diffusion coefficients, theoretical mean diffusion times, and average tracking times according to glycerol concentration. Mean diffusion times were theoretically determined from a 2D model. Average tracking times include all data runs.

Solution (% Glycerol by Mass)	Diffusion Coefficient ($\mu m^2/s$)	Mean Diffusion Time (s)	Average Tracking Time (s)
100%	0.01	12.1	50.1
72%	0.5	0.243	3.20
63%	1.0	0.121	0.739
50%	2.0	0.061	0.435
37%	4.0	0.030	0.097
0%	12.2	0.010	0.078

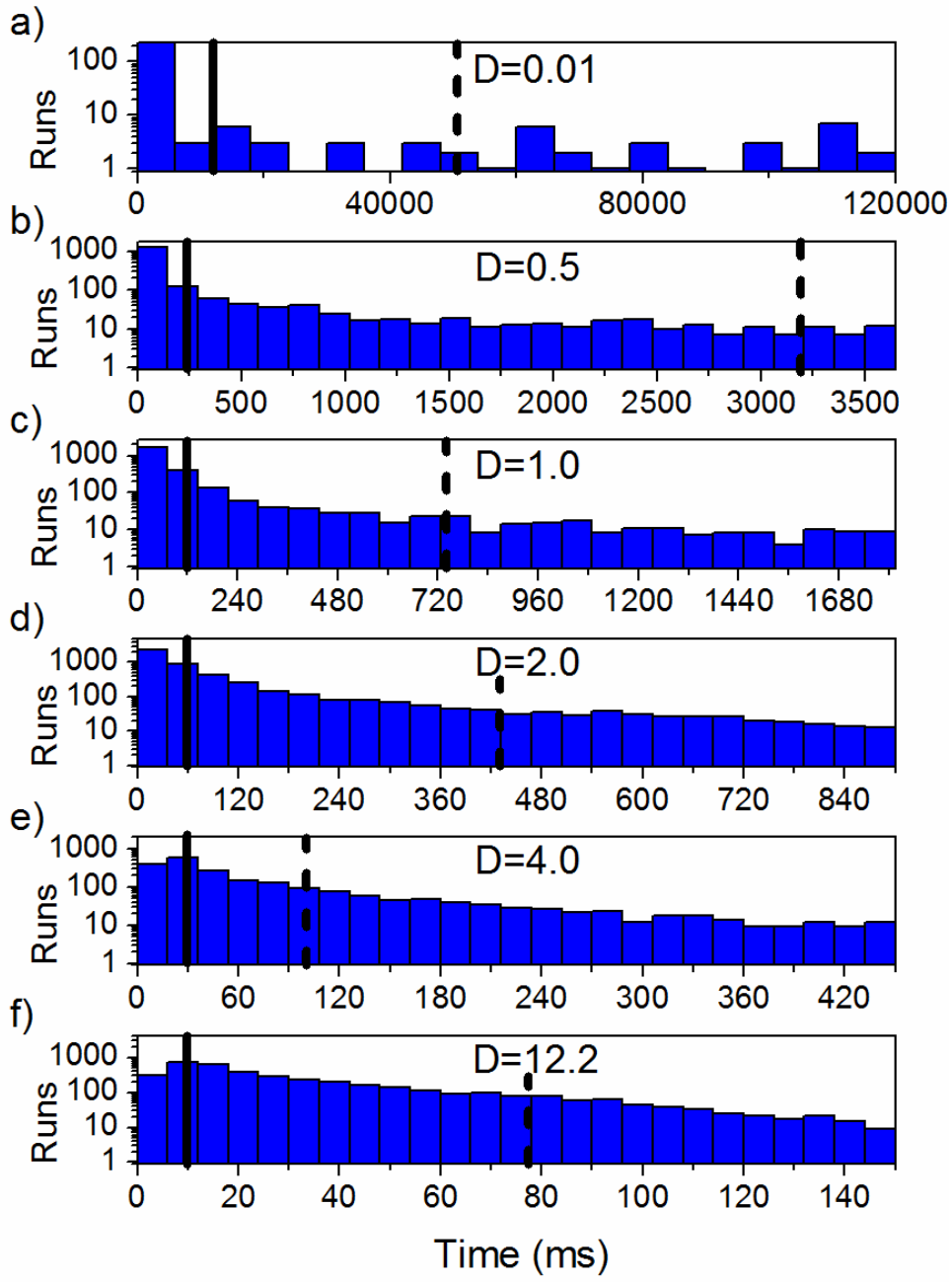


Figure 26. Number of runs vs. total tracking times for various solutions with a) 100% glycerol, b) 72% glycerol by mass, c) 63% glycerol by mass, d) 50% glycerol by mass, e) 37% glycerol by mass, and f) distilled water. The time axis in each case extends to 15 times the mean diffusion time except for 100% glycerol, which extends out to two minutes. Solid lines denote the mean diffusion time and the dotted line denotes the average tracking time. The diffusion coefficients are listed with each graph with units of μm^2/s.

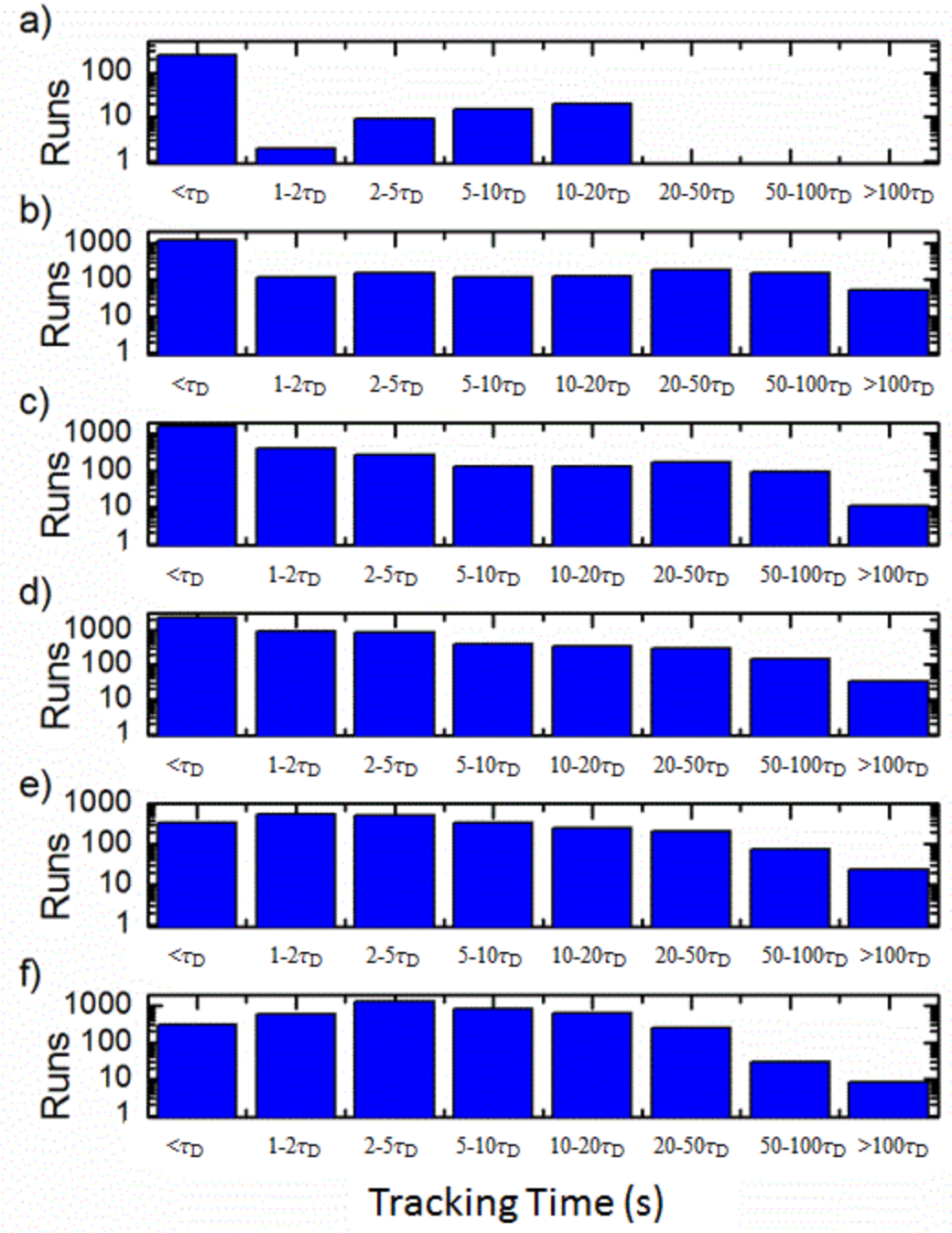


Figure 27. Number of runs vs. tracking time in terms of mean diffusion time for solutions of a) 100% glycerol, b) 72% glycerol by mass, c) 63% glycerol by mass, d) 50% glycerol by mass, e) 37% glycerol by mass, and f) distilled water. The mean diffusion times, τ_D, are those listed in Table 9.

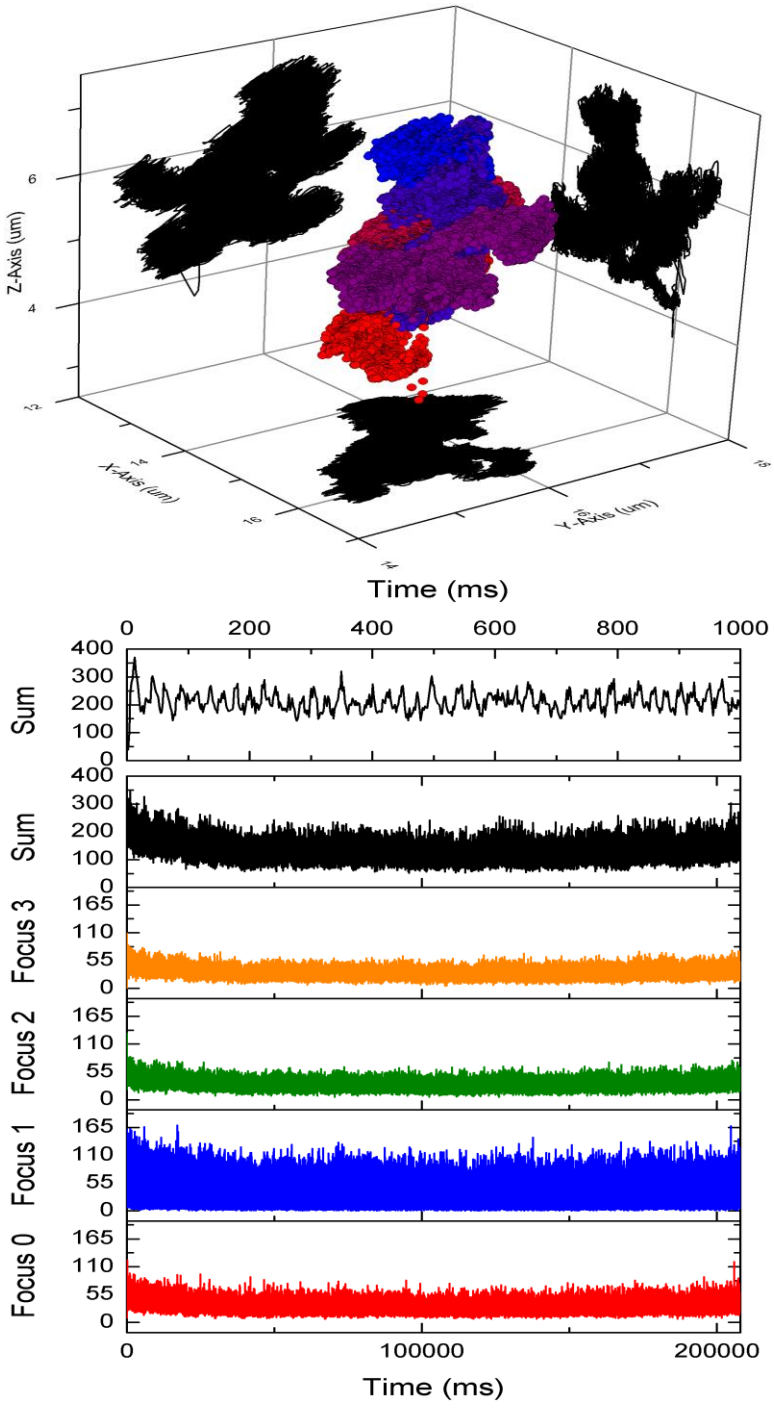


Figure 28. Tracking run of 40 nm fluorescently labeled latex bead in pure glycerol. a) 3D plot of recorded positions taken during the run. The beginning of the run is marked with red symbols and the end of the run is marked with blue symbols. b) Photons counts recorded at each position. The top line chart is the first second of data. Subsequent lines are the total number of counts for all foci and the individual count rates for focus 3, 2, 1, and 0 respectively. This particular tracking run lasted over three minutes.

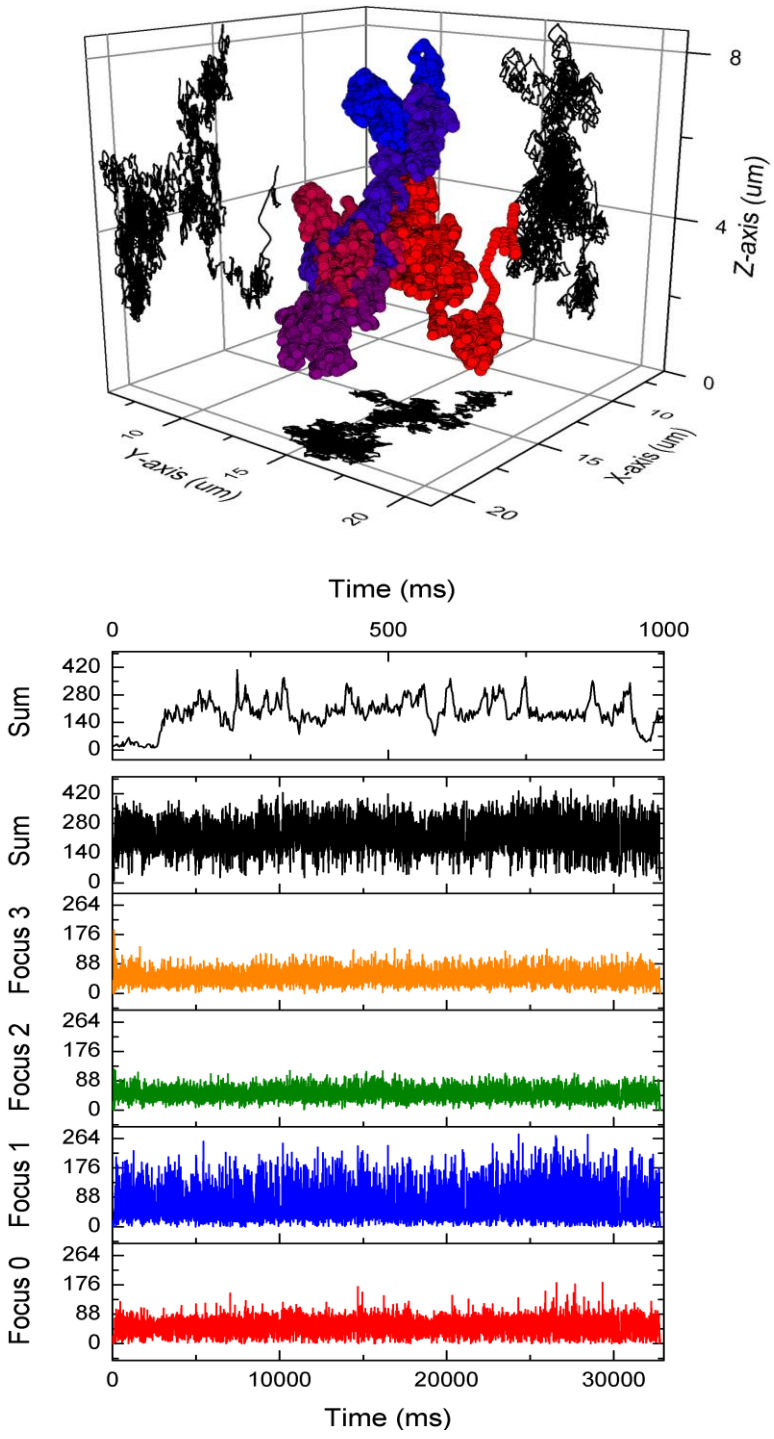


Figure 29. Tracking run of 40 nm fluorescently labeled latex bead 72% glycerol by mass solution. a) 3D plot of recorded positions taken during the run. The beginning of the run is marked with red symbols and the end of the run is marked with blue symbols. b) Photons counts recorded at each position. The top line chart is the first second of data. Subsequent lines are the total number of counts for all foci and the individual count rates for focus 3, 2, 1, and 0 respectively.

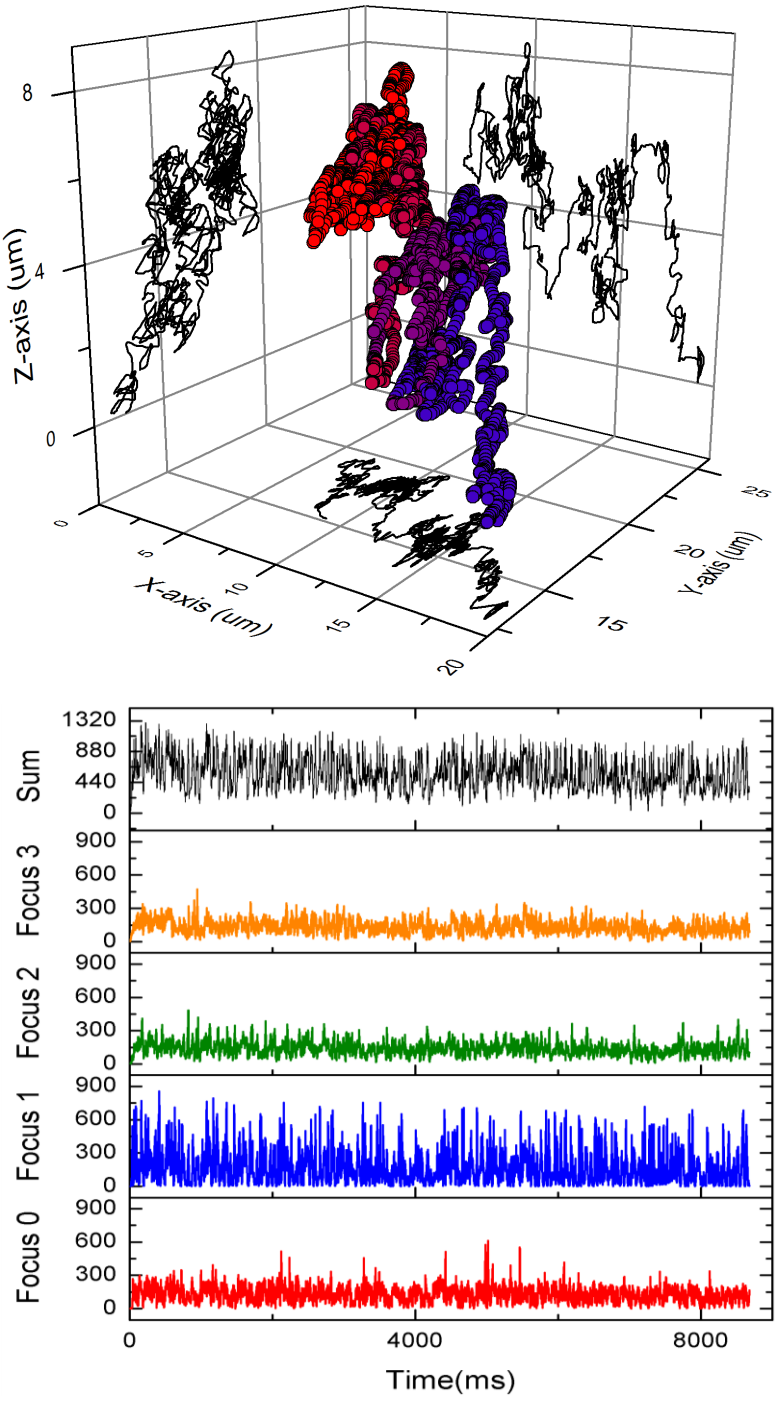


Figure 30. Tracking run of 40 nm fluorescently labeled latex bead in 63% glycerol by mass solution. a) 3D plot of recorded positions taken during the run. The beginning of the run is marked with red symbols and the end of the run is marked with blue symbols. b) Photons counts recorded at each position. The top line chart is the first second of data. Subsequent lines are the total number of counts for all foci and the individual count rates for focus 3, 2, 1, and 0 respectively.

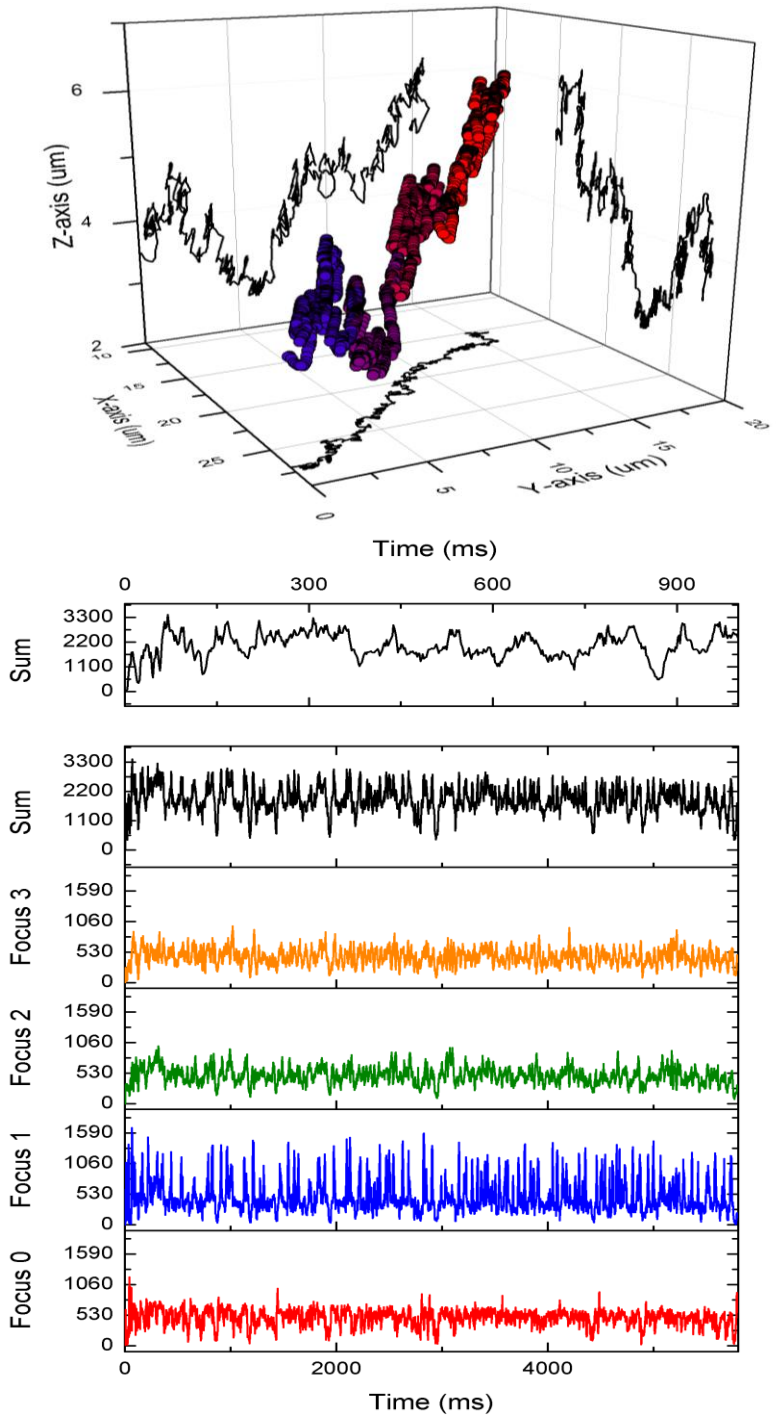


Figure 31. Tracking run of 40 nm fluorescently labeled latex bead in 50% glycerol by mass solution. a) 3D plot of recorded positions taken during the run. The beginning of the run is marked with red symbols and the end of the run is marked with blue symbols. b) Photons counts recorded at each position. The top line chart is the first second of data. Subsequent lines are the total number of counts for all foci and the individual count rates for focus 3, 2, 1, and 0 respectively.

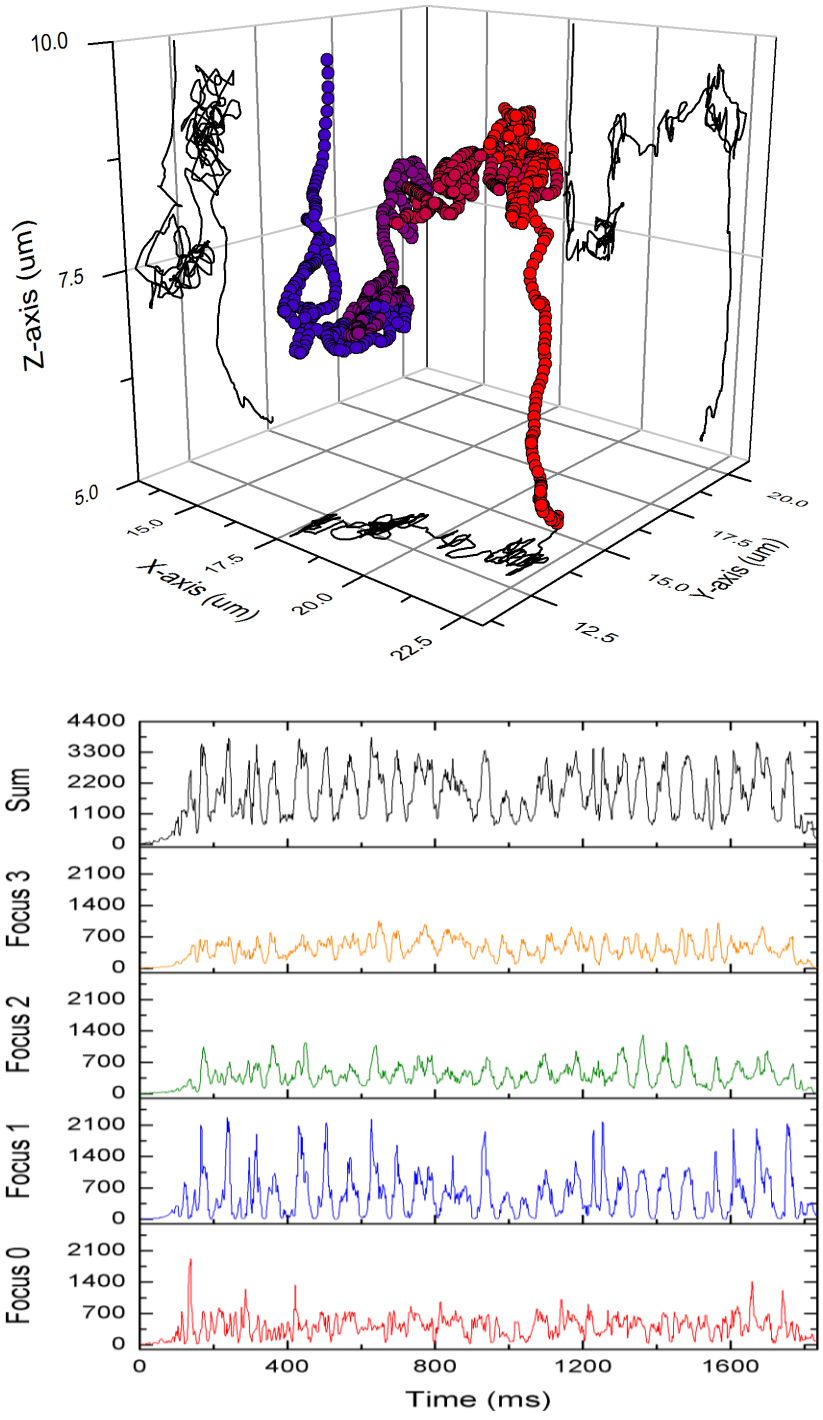


Figure 32. Tracking run of 40 nm fluorescently labeled latex bead in 37% glycerol by mass solution. a) 3D plot of recorded positions taken during the run. The beginning of the run is marked with red symbols and the end of the run is marked with blue symbols. b) Photons counts recorded at each position. The total number of counts for all foci and the individual count rates for focus 3, 2, 1, and 0 respectively are displayed.

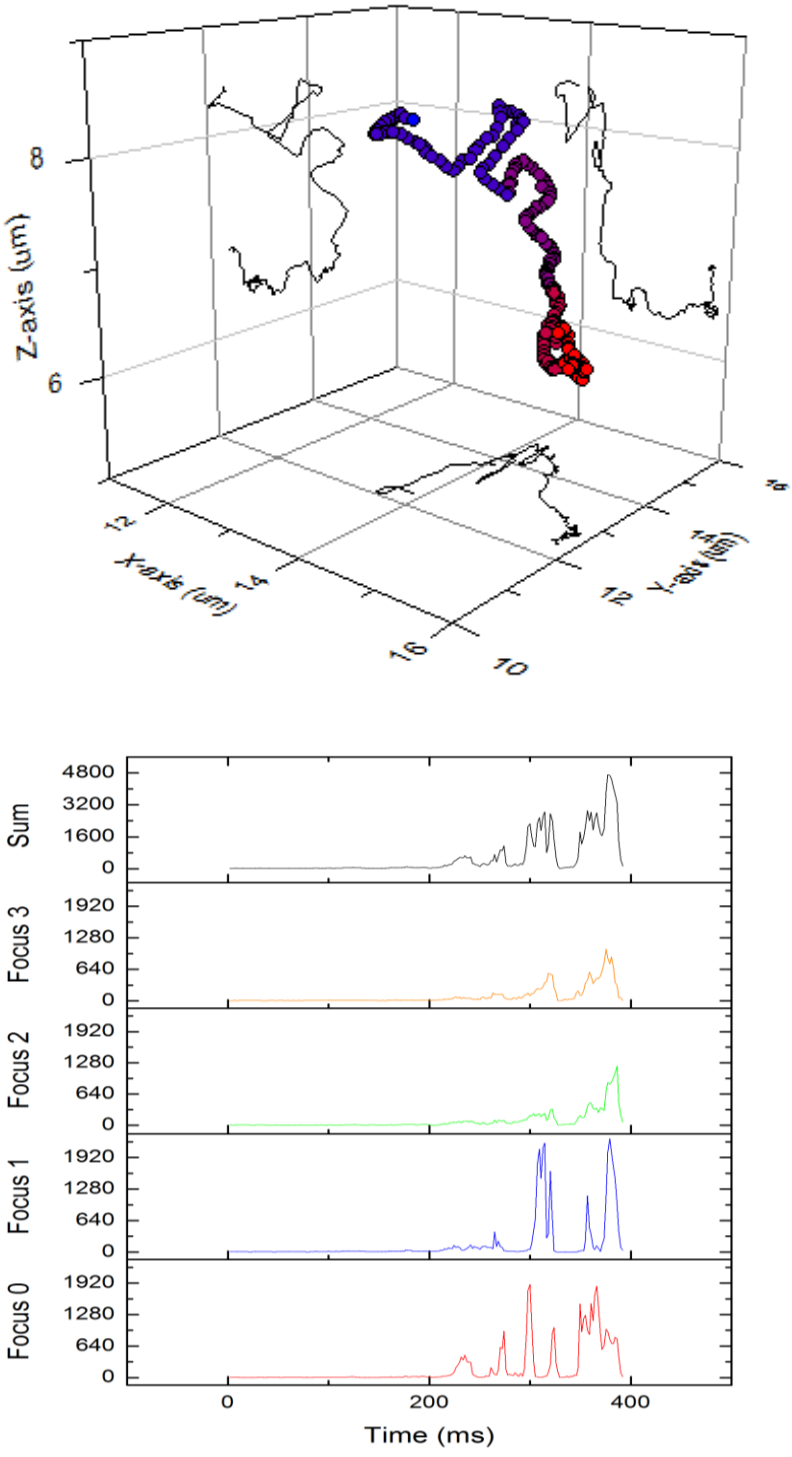


Figure 33. Tracking run of 40 nm fluorescently labeled latex bead in aqueous solution. a) 3D plot of recorded positions taken during the run. The beginning of the run is marked with red symbols and the end of the run is marked with blue symbols. b) Photons counts recorded at each position. The total number of counts for all foci and the individual count rates for focus 3,2,1, and 0 respectively are displayed.

CHAPTER IX-TRACKING DISCUSSION

9.1 Precision of Position Measurements

9.1.1 Precision of X-Axis and Y-Axis Measurements

While we have previously described a method for more precisely estimating the position by maximum likelihood techniques,[36] the position estimated via the linear approximation adopted in equations 7.1a-7.1c are considerably faster to calculate and hence more suitable for real-time applications such as tracking. In this section, the precision of the measurements within the linear approximation are discussed, also assuming that background counts are negligible. Equations 7.1a and 7.1b give estimates for the position along the *x*- and *y*-axes. For the *x*-axis, with the distances from Table 7, the position becomes

$$x = \frac{N_0 \Delta x_0 + N_1 \Delta x_1}{N_{tot}}. \qquad 9.1$$

The focal position is assumed to be constant, and the error in photon counts is assumed to be Poissonian and so equal to the square root of photon counts. The root mean square (RMS) error in position (x_ε) is then

$$x_\varepsilon = \sqrt{\frac{\left(\overline{N_0}\,\Delta x_0 + \overline{N_1}\,\Delta x_1\right)^2}{N_{tot}^2} + \frac{\left(N_0 \Delta x_0 + N_1 \Delta x_1\right)^2}{N_{tot}^3}}. \qquad 9.2$$

Similarly for the *y*-axis, the error in position (y_ε) is

$$y_\varepsilon = \sqrt{\frac{\left(\overline{N_2}\,\Delta y_2 + \overline{N_3}\,\Delta y_3\right)^2}{N_{tot}^2} + \frac{\left(N_2 \Delta y_2 + N_3 \Delta y_3\right)^2}{N_{tot}^3}}. \qquad 9.3$$

The lowest error occurs when the number of counts for each focus is the same, which is when the bead is in the center of the tetrahedron. For a threshold of 10 total counts and an average of 2.5 counts from each focus, the error in position at the center of the tetrahedron in both *x* and *y* is 7.9 nm. The error for position estimates away the center is larger. For example, if $N_0 = 9$, $N_1 = 1$, and $N_{tot} = 20$, a count rate representative of a bead located toward the edge of the tetrahedron, then the error in *x* is 87 nm.

9.1.2 Precision of Z-Axis Measurements

Measurement of precision along the *z*-axis is similar to the *x*-axis and *y*-axis cases. The *z*-axis precision uses all four foci, though. The RMS error here, as computed from equation 6.1c, is

$$z_\varepsilon = \frac{\left(\overline{N_0}\,\Delta z_0 + \overline{N_1}\,\Delta z_1 + \overline{N_2}\,\Delta z_2 \overline{N_3}\,\Delta z_3\right)^2}{N_{tot}^2} + \frac{\left(N_0\Delta z_0 + N_1\Delta z_1 + N_2\Delta z_{2+} N_3\Delta z_3\right)^2}{N_{tot}^3}^{0.5}. \quad 9.4$$

The error along the optical axis for a measurement where the total photon count is at threshold (10 counts) and the average number of counts per focus set at 2.5 is 61.4 nm. The increased size of error in the *z*-axis case arises from the larger distance between foci along the optical axis. Of course, as the total count rate rises, the error decreases.

9.2 Response Parameters

Initial response parameter values were determined empirically by following a latex bead in glycerol. The parameters were adjusted until the piezoelectric stage displacement was

minimized. Parameters in all data sets were initially set at $K_x = K_y = 0.8$ and $K_z = 0.165$. The parameters were varied slightly (0.0–1.5) without noticeable changes in the tracking times, the ratio of particles lost, or the total number of tracking runs. An oscillation in the total count rate is sometimes observed, for example in Figure 32. Such oscillations can arise in a control system with latency and may be reduced by reducing the feedback gain. Improved tracking results could possibly be obtained with a formalized study of tracking time versus response parameter values and diffusion coefficients. Fortunately, the tracking setup and algorithm were robust enough to provide convincing results of tracking without such a study.

9.3 Recommendations for Future Experiments

Latex beads were tracked freely diffusing in solution. These beads encapsulate ~300 dye molecules in their centers, making the beads relatively bright. The next step is to trap smaller, dimmer particles. A good choice of particle is Alexa 647 labeled streptavidin, which has three Alexa molecules per streptavidin protein. The diffusion coefficient of streptavidin is an order of magnitude larger in water, 130 ± 10 $\mu m^2/s$,[71] and thus requires a more viscous (higher glycerol content) solution to reduce the diffusion coefficient to 0.5 or 1.0 $\mu m^2/s$. For these lower diffusion coefficients, 0.5 and 1.0 $\mu m^2/s$, the tracking routine performed admirably. Average occupation time is increased by at least a factor of 5. These diffusivities are equivalent to a particle moving in the membrane of a cell, [68] making the four-focus microscope a useful tool for studying the mechanics of cells.

The loop time of 1.84 ms in the LabVIEW program limits the temporal resolution of position measurements. The bottleneck in the tracking program is the speed of operation of the piezoelectric stage, including the communication architecture between the Target PC and the

piezo stage controller, which requires resetting 26 digital voltage lines a number of times for each command. The diodes and SPADs by themselves are able to achieve loop times of close to 200 μs. Instead of a piezoelectric stage compensating for diffusional movement, a trapping setup could be introduced. A candidate for 3D trapping is the Anti-Brownian Electrokentic (ABEL) setup[72], which uses a controllable electric field to direct flow in a sample. This would eliminate the physical limitation of movement and would require fewer digital lines.

In comparison to other confocal tracking setups, the four-focus microscope provides faster 3D position measurements and most probably has higher fluorescence throughput. When compared to the four pinhole method, the response time is approximately the same (~2 ms) as the same piezo stage is used in that work, [68] but the reduced number of apertures allows for more fluorescence throughput in the four-focus setup, especially in the center of the excitation tetrahedron. On the other hand, the four-focus microscope and the revolving focus should have roughly the same fluorescence throughput, but the revolving focus has a slower response time (32 ms).[66] In a sense, the four-focus microscope can have the best of both worlds; fast response time and high fluorescence throughput.

CHAPTER X-CONCLUSIONS

Two possible setups were tested using the centerpiece of measurement of four excitation volumes: four-focus FCCS and four-focus position determination/tracking. Simulations of particles, modeled after Rhodamine B, diffusing and moving with bulk flow were conducted. A fitting program was created to globally fit the 3D flow velocities to 16 correlation curves. Two different models, the 3D Gaussian and Gaussian-Lorentzian models were tested to determine which would serve as the better model. Both models were able to fit one-focus and two-focus systems, and the 3D flow can be measured approximately along the *x*- and *y*-axes. However, the errors are considerable along the *z*-axis.

The four-focus tracking system used four modulated laser diodes to determine position based on the relative fluorescence generated by each focus. Fluorescence intensity and position were recorded and a Python script was used to map out individual runs. The four-focus scheme was able to track fluorescently labeled latex beads using a piezoelectric stage with a precision of ~60 nm along the optical axis and ~ 8 nm in-plane. The efficiency of the four-focus setup is also well within the range of single-molecule sensitivity. The tracking routine was able to extend the average occupation time in the tetrahedral excitation volume by a factor of 5. The occupation time could be increased by replacing the piezoelectric stage with a different device for following or trapping single nanoparticles, such as the electrokinetic trap. As the setup was able to track beads efficiently with diffusion coefficients from 0.5–1.0 $\mu m^2/s$, the four-focus microscope would be a good tool for tracking particles traveling along and through cell membranes.

LIST OF REFERENCES

1. D. Magde, W. W. Webb, and E. Elson, "Thermodynamic fluctuations in a reacting system - measurement by fluorescence correlation spectroscopy," Phys. Rev. Lett. 29, 705-& (1972).
2. D. Magde, E. L. Elson, and W. W. Webb, "Fluorescence correlation spectroscopy .2. Experimental realization," Biopolymers 13, 29-61 (1974).
3. E. L. Elson and D. Magde, "Fluorescence correlation spectroscopy .1. Conceptual basis and theory," Biopolymers 13, 1-27 (1974).
4. D. E. Koppel, D. Axelrod, J. Schlessinger, E. L. Elson, and W. W. Webb, "Dynamics of fluorescence marker concentration as a probe of mobility," Biophys. J. 16, 1315-1329 (1976).
5. D. Magde, W. W. Webb, and E. L. Elson, "Fluorescence correlation spectroscopy .3. Uniform translation and laminar-flow," Biopolymers 17, 361-376 (1978).
6. P. S. Dittrich and P. Schwille, "Spatial two-photon fluorescence cross-correlation Spectroscopy for controlling molecular transport in microfluidic structures," Analytical chemistry 74, 4472-4479 (2002).
7. K. M. Berland, P. T. C. So, and E. Gratton, "2-photon fluorescence correlation spectroscopy - method and application to the intracellular environment," Biophys. J. 68, 694-701 (1995).
8. R. H. Kohler, P. Schwille, W. W. Webb, and M. R. Hanson, "Active protein transport through plastid tubules: velocity quantified by fluorescence correlation spectroscopy," J. Cell Sci. 113, 3921-3930 (2000).
9. T. J. Arbour and J. Enderlein, "Application of dual-focus fluorescence correlation spectroscopy to microfluidic flow-velocity measurement," Lab on a Chip 10, 1286-1292 (2010).
10. H. Qian and E. L. Elson, "Analysis of confocal laser-microscope optics for 3-D fluorescence correlation spectroscopy," Appl. Optics 30, 1185-1195 (1991).
11. J. R. Lakowicz, *Principles of fluorescence spectroscopy*, 3rd ed. (Springer, New York, 2006), pp. xxvi, 954 p.
12. P. T. C. So, H. Kim, and I. E. Kochevar, "Two-photon deep tissue ex vivo imaging of mouse dermal and subcutaneous structures," Opt. Express 3, 339-350 (1998).
13. R. H. Webb, "Confocal optical microscopy," Rep. Prog. Phys. 59, 427-471 (1996).
14. D. R. Sandison and W. W. Webb, "Background rejection and signal-to-noise optimization in confocal and alternative fluorescence microscopes," Appl. Optics 33, 603-615 (1994).
15. J. Korlach, P. Schwille, W. W. Webb, and G. W. Feigenson, "Characterization of lipid bilayer phases by confocal microscopy and fluorescence correlation spectroscopy," Proc. Natl. Acad. Sci. U. S. A. 96, 8461-8466 (1999).
16. D. R. Sisan, R. Arevalo, C. Graves, R. McAllister, and J. S. Urbach, "Spatially resolved fluorescence correlation spectroscopy using a spinning disk confocal microscope," Biophys. J. 91, 4241-4252 (2006).
17. K. Bacia, S. A. Kim, and P. Schwille, "Fluorescence cross-correlation spectroscopy in living cells," Nat. Methods 3, 83-89 (2006).

18. **W. Wiegraebe, "Two-Photon Microscopy, Non-Linear Optics (NLO)", retrieved 5-1-13, 2013, http://research.stowers-institute.org/microscopy/external/Technology/NLO/index.htm.**
19. **W. Denk, "2-photon scanning photochemical microscopy - Mapping ligand-gated ion-channel distributions," Proc. Natl. Acad. Sci. U. S. A. 91, 6629-6633 (1994).**
20. **C. Xu and W. W. Webb, "Measurement of two-photon excitation cross sections of molecular fluorophores with data from 690 to 1050 nm," J. Opt. Soc. Am. B-Opt. Phys. 13, 481-491 (1996).**
21. **P. T. C. So, C. Y. Dong, B. R. Masters, and K. M. Berland, "Two-photon excitation fluorescence microscopy," Annu. Rev. Biomed. Eng. 2, 399-429 (2000).**
22. **W. Denk, J. H. Strickler, and W. W. Webb, "2-photon laser scanning fluorescence microscopy," Science 248, 73-76 (1990).**
23. **M. Gu and C. J. R. Sheppard, "Effects of a finite-sized pinhole 3D image-formation in confocal 2-photon fluorescence microscopy," J. Mod. Opt. 40, 2009-2024 (1993).**
24. **R. Gauderon, P. B. Lukins, and C. J. R. Sheppard, "Effect of a confocal pinhole in two-photon microscopy," Microsc. Res. Tech. 47, 210-214 (1999).**
25. **Y. Sako, A. Sekihata, Y. Yanagisawa, M. Yamamoto, Y. Shimada, K. Ozaki, and A. Kusumi, "Comparison of two-photon excitation laser scanning microscopy with UV-confocal laser scanning microscopy in three-dimensional calcium imaging using the fluorescence indicator Indo-1," J. Microsc.-Oxf. 185, 9-20 (1997).**
26. **E. Gratton, N. P. Barry, S. Beretta, and A. Celli, "Multiphoton fluorescence microscopy," Methods 25, 103-110 (2001).**
27. **M. Rubart, "Two-photon microscopy of cells and tissue," Circ.Res. 95, 1154-1166 (2004).**
28. **N. P. Barry, K. M. Hanson, E. Gratton, R. M. Clegg, M. J. Behne, and T. M. Mauro, "Applications of ultrafast lasers to two-photon fluorescence and lifetime imaging," in *Commercial and Biomedical Applications of Ultrafast and Free-Electron Lasers*, G. S. Edwards, J. Neev, A. Ostendorf, and J. C. Sutherland, eds. (Spie-Int Soc Optical Engineering, Bellingham, 2002), pp. 50-61.**
29. **M. J. Levene, D. A. Dombeck, K. A. Kasischke, R. P. Molloy, and W. W. Webb, "In vivo multiphoton microscopy of deep brain tissue," J. Neurophysiol. 91, 1908-1912 (2004).**
30. **G. D. J. Phillies, "Suppression of multiple-scattering effects in quasi-elastic light-scattering by homodyne cross-correlation techniques," J. Chem. Phys. 74, 260-262 (1981).**
31. **P. Schwille, F. J. MeyerAlmes, and R. Rigler, "Dual-color fluorescence cross-correlation spectroscopy for multicomponent diffusional analysis in solution," Biophys. J. 72, 1878-1886 (1997).**
32. **X. Michalet, R. A. Colyer, G. Scalia, T. Kim, M. Levi, D. Aharoni, A. Cheng, F. Guerrieri, K. Arisaka, J. Millaud, I. Rech, D. Resnati, S. Marangoni, A. Gulinatti, M. Ghioni, S. Tisa, F. Zappa, S. Cova, and S. Weiss, "High-throughput single-molecule fluorescence spectroscopy using parallel detection," 76082D-76082D (2010).**

33. N. P. Wells, G. A. Lessard, and J. H. Werner, "Confocal, Three-Dimensional Tracking of Individual Quantum Dots in High-Background Environments," Analytical chemistry 80, 9830-9834 (2008).
34. P. Didier, J. Godet, and Y. Mely, "Two-Photon Two-Focus Fluorescence Correlation Spectroscopy with a Tunable Distance Between the Excitation Volumes," J. Fluoresc. 19, 561-565 (2009).
35. W. E. Moerner and D. P. Fromm, "Methods of single-molecule fluorescence spectroscopy and microscopy," Rev. Sci. Instrum. 74, 3597-3619 (2003).
36. L. M. Davis, B. K. Canfield, J. A. Germann, J. K. King, W. N. Robinson, A. D. Dukes, III , S. J. Rosenthal, P. C. Samson, and J. P. Wikswo, "Four-focus single-particle position determination in a confocal microscope," Proc. of SPIE 7571(2010).
37. W. N. Robinson and L. M. Davis, "Simulation of single-molecule trapping in a nanochannel," Journal of Biomedical Optics 15, 045006-045006-045012 (2010).
38. M. Marrocco, "Two-photon excitation fluorescence correlation Spectroscopy of diffusion for Gaussian-Lorentzian volumes," J. Phys. Chem. A 112, 3831-3836 (2008).
39. M. Y. Berezin and S. Achilefu, "Fluorescence Lifetime Measurements and Biological Imaging," Chem. Rev. 110, 2641-2684 (2010).
40. T. Ha, T. Enderle, D. F. Ogletree, D. S. Chemla, P. R. Selvin, and S. Weiss, "Probing the interaction between two single molecules: Fluorescence resonance energy transfer between a single donor and a single acceptor," Proc. Natl. Acad. Sci. U. S. A. 93, 6264-6268 (1996).
41. I. Sase, H. Miyata, S. Ishiwata, and K. Kinosita, "Axial rotation of sliding actin filaments revealed by single-fluorophore imaging," Proc. Natl. Acad. Sci. U. S. A. 94, 5646-5650 (1997).
42. S. J. Attwood, A. M. C. Simpson, R. Stone, S. W. Hamaia, D. Roy, R. W. Farndale, M. Ouberai, and M. E. Welland, "A Simple Bioconjugate Attachment Protocol for Use in Single Molecule Force Spectroscopy Experiments Based on Mixed Self-Assembled Monolayers," Int. J. Mol. Sci. 13, 13521-13541 (2012).
43. N. F. Reuel, A. Dupont, O. Thouvenin, D. C. Lamb, and M. S. Strano, "Three-Dimensional Tracking of Carbon Nanotubes within Living Cells," ACS Nano 6, 5420-5428 (2012).
44. W. E. Moerner, "New directions in single-molecule imaging and analysis," Proc. Natl. Acad. Sci. U. S. A. 104, 12596-12602 (2007).
45. P. R. Smith, I. E. G. Morrison, K. M. Wilson, N. Fernandez, and R. J. Cherry, "Anomalous diffusion of major histocompatibility complex class I molecules on HeLa cells determined by single particle tracking," Biophys. J. 76, 3331-3344 (1999).
46. I. Izeddin, M. El Beheiry, J. Andilla, D. Ciepielewski, X. Darzacq, and M. Dahan, "PSF shaping using adaptive optics for three-dimensional single-molecule super-resolution imaging and tracking," Opt. Express 20, 4957-4967 (2012).
47. J. H. Spille, T. Kaminski, H. P. Konigshoven, and U. Kubitscheck, "Dynamic three-dimensional tracking of single fluorescent nanoparticles deep inside living tissue," Opt. Express 20, 19697-19707 (2012).

48. **M. K. Cheezum, W. F. Walker, and W. H. Guilford, "Quantitative comparison of algorithms for tracking single fluorescent particles," Biophys. J. 81, 2378-2388 (2001).**

49. **C. M. Anderson, G. N. Georgiou, I. E. G. Morrison, G. V. W. Stevenson, and R. J. Cherry, "Tracking of cell-surface receptors by fluorescence digital imaging microscopy using a charge-coupled device camera - Low density lipoprotein and influenza virus receptor mobility at 4 degrees C," J. Cell Sci. 101, 415-425 (1992).**

50. **S. Ram, P. Prabhat, J. Chao, E. S. Ward, and R. J. Ober, "High Accuracy 3D Quantum Dot Tracking with Multifocal Plane Microscopy for the Study of Fast Intracellular Dynamics in Live Cells," Biophys. J. 95, 6025-6043 (2008).**

51. **M. Bates, B. Huang, and X. W. Zhuang, "Super-resolution microscopy by nanoscale localization of photo-switchable fluorescent probes," Curr. Opin. Chem. Biol. 12, 505-514 (2008).**

52. **S. R. P. Pavani, M. A. Thompson, J. S. Biteen, S. J. Lord, N. Liu, R. J. Twieg, R. Piestun, and W. E. Moerner, "Three-dimensional, single-molecule fluorescence imaging beyond the diffraction limit by using a double-helix point spread function," Proc. Natl. Acad. Sci. U. S. A. 106, 2995-2999 (2009).**

53. **D. R. Sandison, D. W. Piston, R. M. Williams, and W. W. Webb, "Quantitative comparison of background rejection, signal-to-noise ratio, and resolution in confocal and full-field laser scanning microscopes," Appl. Optics 34, 3576-3588 (1995).**

54. **M. Bohmer and J. Enderlein, "Fluorescence spectroscopy of single molecules under ambient conditions: Methodology and technology," ChemPhysChem 4, 793-808 (2003).**

55. **A. Diaspro and M. Robello, "Two-photon excitation of fluorescence for three-dimensional optical imaging of biological structures," J. Photochem. Photobiol. B-Biol. 55, 1-8 (2000).**

56. **X. Michalet, O. H. W. Siegmund, J. V. Vallerga, P. Jelinsky, J. E. Millaud, and S. Weiss, "A space- and time-resolved single photon counting detector for fluorescence microscopy and spectroscopy," 60920M-60920M (2006).**

57. **J. R. Unruh and E. Gratton, "Analysis of Molecular Concentration and Brightness from Fluorescence Fluctuation Data with an Electron Multiplied CCD Camera," Biophys. J. 95, 5385-5398 (2008).**

58. **C. R. Sheppard, X. Gan, M. Gu, and M. Roy, "Signal-to-Noise in Confocal Microscopes," in *Handbook of Biological Confocal Microscopy*, J. Pawley, ed. (Springer US, 1995), pp. 363-371.**

59. **E. H. K. Stelzer, "Light microscopy: Beyond the diffraction limit?," Nature 417, 806-807 (2002).**

60. **X. Zhang and Z. W. Liu, "Superlenses to overcome the diffraction limit," Nat. Mater. 7, 435-441 (2008).**

61. **E. Boukobza, A. Sonnenfeld, and G. Haran, "Immobilization in surface-tethered lipid vesicles as a new tool for single biomolecule spectroscopy," J. Phys. Chem. B 105, 12165-12170 (2001).**

62. **D. V. Patel and C. N. J. McGhee, "Mapping the corneal sub-basal nerve plexus in keratoconus by in vivo laser scanning confocal microscopy," Invest. Ophthalmol. Vis. Sci. 47, 1348-1351 (2006).**
63. **M. D. Burke, J. O. Park, M. Srinivasarao, and S. A. Khan, "Diffusion of macromolecules in polymer solutions and gels: A laser scanning confocal microscopy study," Macromolecules 33, 7500-7507 (2000).**
64. **K. Koenig and I. Riemann, "High-resolution multiphoton tomography of human skin with subcellular spatial resolution and picosecond time resolution," Journal of Biomedical Optics 8, 432-439 (2003).**
65. **K. Kis-Petikova and E. Gratton, "Distance measurement by circular scanning of the excitation beam in the two-photon microscope," Microsc. Res. Tech. 63, 34-49 (2004).**
66. **V. Levi, Q. Q. Ruan, and E. Gratton, "3-D particle tracking in a two-photon microscope: Application to the study of molecular dynamics in cells," Biophys. J. 88, 2919-2928 (2005).**
67. **G. A. Lessard, P. M. Goodwin, and J. H. Werner, "Three-dimensional tracking of individual quantum dots," Appl. Phys. Lett. 91(2007).**
68. **J. J. Han, C. Kiss, A. R. M. Bradbury, and J. H. Werner, "Time-Resolved, Confocal Single-Molecule Tracking of Individual Organic Dyes and Fluorescent Proteins in Three Dimensions," ACS Nano 6, 8922-8932 (2012).**
69. **N. P. Wells, G. A. Lessard, M. E. Phipps, P. M. Goodwin, D. S. Lidke, B. S. Wilson, and J. H. Werner, "Going beyond 2D: following membrane diffusion and topography in the IgE-Fc[epsilon]RI system using 3-dimensional tracking microscopy," 71850Z-71850Z (2009).**
70. **N. S. Cheng, "Formula for the viscosity of a glycerol-water mixture," Ind. Eng. Chem. Res. 47, 3285-3288 (2008).**
71. **L. Zhang, K. Dammann, S. C. Bae, and S. Granick, "Ligand-receptor binding on nanoparticle-stabilized liposome surfaces," Soft Matter 3, 551-553 (2007).**
72. **J. K. King, B. K. Canfield, and L. M. Davis, "Three-dimensional anti-Brownian electrokinetic trapping of a single nanoparticle in solution," Appl. Phys. Lett. 103, 043102 (2013).**

APPENDIX

Appendix A. Python Fitting Code

This code analyzes text spreadsheets of correlation functions. Time, correlation, and weight are read in from the spreadsheet. A least squares package is used to fit the data. The correlation functions are weighted for the fit using the square root counted photon pairs. The provided code fits for either the 3D Gaussian model or Gaussian-Lorentzian model. The code also fits one, two, or four foci. Numerical integration for the Gaussian-Lorentzian code is done with Simpson's rule and also with an inline call to C for quicker execution speed.

A.1 One-Focus 3D Gaussian Code without Triplet Fitting

```
import time
import numpy as np
from scipy.optimize import leastsq
import pylab as plt

#These are the constants for the foci and physical system
z_r = 0.98  #Optical axis beam waist
w_0 = 0.3535534  #Beam waist-modified for two-photon excitation
D = 220 #Diffusion coefficient for Rhodamine b
p0 = [100, 5000]  #Fitting parameters


func = lambda t, Tvx, Tvy, Tvz, Td, deltax, deltay, deltaz:  ((1+t/Td)**-
1)*((np.sqrt(1+((w_0/z_r)**2)*t/Td))**-1)*np.exp(-((t/Tvx-deltax/w_0)**2)/(1+t/Td))*np.exp(-
((t/Tvy-deltay/w_0)**2)/(1+t/Td))*np.exp(-((t/Tvz-deltaz/z_r)**2)/(1+((w_0/z_r)**2)*t/Td))


f_name = "c:/users/jgermann/desktop/python/data/Oct2012runs/1focus20mmpersnotrip.xls"
f = open(f_name, 'r')

data = []

for line in f:
    data.append(line.split('\t'))

f.close()
```

```
data = np.array(data)

sp = 6   #Start at point 6, or 1 microsecond on the spreadsheet
td = data[sp:,0].astype(float)
autofunc = data[sp:,1].astype(float)
weight = data[sp:,17].astype(float)

fitfunc = lambda p, xx: (1+p[0]*func(xx, w_0/p[1], 1e21, 1e21, (w_0**2)/(4*D), 0, 0, 0))
errfunc = lambda p, xx, y, w: ((fitfunc(p,xx)-y)*(np.sqrt(w))**-1)

start = time.time()
plsq = leastsq(errfunc, p0[:], args = (td, autofunc, weight), maxfev=5000000)
end = time.time()

print plsq

#Plotting code
fig = plt.figure(0, (9,6))
ax = fig.add_subplot(1,1,1)
ax.set_xscale('log')
ax.set_ylim(bottom=0, top=2)
ax.set_xlim(left=5e-7)
plt.scatter(td,autofunc,color = 'red', marker = '^')
#plt.plot(td, fitfunc(plsq[0],td), color = 'blue', lw = 2)
plt.grid()
##plt.title(f_name + '\t$A = %5.4f | V_x = %0.4f$' % (plsq[0][0], plsq[0][1]))
plt.show()
#Plotting code end
print ((end-start)/60) #Used timing purposes
```

A.2 One-Focus 3D Gaussian Code with Triplet Fitting

```
import time
import numpy as np
from scipy.optimize import leastsq
import pylab as plt

#These are the constants for the foci and physical system
z_r = 0.98 #Rayleigh Range
w_0 = 0.3535534 #Beam waist, adjusted for two-photon excitation
D = 220 #Diffusion coefficent
p0 = [1000, 100000, 1.0, 1.0, 1.0e-3] #Best guess
triplife = 1.0e-6 #Triplet lifetime in units of seconds
probtrip = 1.0e-3 #Odds of molecule being in triplet state
```

```
func = lambda t, Tvx, Tvy, Tvz, Td, deltax, deltay, deltaz, pt1, pt2, tl:  (pt1+pt2*np.exp(-
t/tl))*(((1+t/Td)**-1)*((np.sqrt(1+((w_0/z_r)**2)*t/Td))**-1)*np.exp(-((t/Tvx-
deltax/w_0)**2)/(1+t/Td))*np.exp(-((t/Tvy-deltay/w_0)**2)/(1+t/Td))*np.exp(-((t/Tvz-
deltaz/z_r)**2)/(1+((w_0/z_r)**2)*t/Td)))

f_name = "c:/users/jgermann/desktop/python/data/100 mmpers seed 2011.xls"
f = open(f_name, 'r')

data = []

for line in f:
   data.append(line.split('\t'))

f.close()

data = np.array(data)

sp = 6
td = data[sp:,0].astype(float)
autofunc = data[sp:,1].astype(float)
weight = data[sp:,17].astype(float)

fitfunc = lambda p, xx: (1+p[0]*func(xx, w_0/p[1], 1e21, 1e21, (w_0**2)/(4*D), 0, 0, 0, p[2],
p[3], p[4]))
errfunc = lambda p, xx, y, w: ((fitfunc(p,xx)-y)*(np.sqrt(w))**-1)

start = time.time()
plsq = leastsq(errfunc, p0[:], args = (td, autofunc, weight), maxfev = 5000000)
end = time.time()

print plsq[0]
#Plot data begin
fig = plt.figure(0, (9,6))
ax = fig.add_subplot(1,1,1)
ax.set_xscale('log')
ax.set_ylim(bottom=0, top=(max(autofunc)+10))
ax.set_xlim(left=5e-7)
plt.scatter(td,autofunc,color = 'red', marker = '^')
plt.plot(td, fitfunc(plsq[0],td), color = 'blue', lw = 2)
plt.grid()
##plt.title(f_name + '\t$A = %5.4f | V_x = %0.4f$' % (plsq[0][0], plsq[0][1]))
plt.show()
#Plot data end
print ((end-start)/60)
```

A.3 One-Focus Gaussian-Lorentzian Code without Triplet Fitting

```
import numpy as np
import pylab as plt
from scipy.optimize import leastsq
import scipy.weave as weave
import time

errfunc = lambda p, x, t, w:  np.abs((test_func(p,t)-x)*np.sqrt(w)**-1)

f = open('c:/Users/jgermann/Desktop/Python/Data/100 mmpers seed 2011.xls', 'r')
dump = []
for line in f:
   dump.append(line.split('\t'))

sp = 6
dump = np.array(dump)
t = dump[sp:,0].astype('float')
x = dump[sp:,1].astype('float')
weights = dump[sp:,17].astype('float')

code = \
   r'''
   double total = 0.;
   double h_x = 2./(N-1);
   double h_y = 2./(N-1);
   double x; double y; double w;
   int i; int j;


   for(i=1;i<=N+1;i++)
      {
      for(j=1;j<=N+1;j++)
         {
         x = i*h_x;
         y = j*h_y;
         w = weight(i,j,N);
         total += w*func(x,y,po[1],t);
         }
      }
   total *= 4./(9.*N*N);
   total = 1+((po[0])*total/sqrt(t));
   return_val = total;
   '''
```

```
support = \
   r'''
   double weight( int i, int j, int N )
      {
      if(i==1 || i==N+1)
         {
         if(j==1 || j==N+1) return(1.);
         if(j%2==0) return(4.);
         else return(2.);
         }
      if(i%2==0)
         {
         if(j==N+1) return(4.);
         if(j%2==0) return(16.);
         else return(8.);
         }
      else
         {
         if(j==N+1) return(2.);
         if(j%2==0) return(8.);
         else return(4.);
         }
      }

   double func( double z_1, double z_2, double v_x, double t)
      {
      double z_r = 0.98; double w_0 = 0.5; double D = 220.0;
      double deltax = 0.0;
      double wsqr_1 = (w_0*w_0)*(1+(z_1/z_r)*(z_1/z_r));
      double wsqr_2 = (w_0*w_0)*(1+(z_2/z_r)*(z_2/z_r));
      return((1/(wsqr_1*wsqr_2))*(1/(2*D*t + wsqr_1/8 + wsqr_2/8))*exp(-((v_x*t-
         deltax)*(v_x*t-deltax))/(4*D*t + wsqr_1/4 + wsqr_2/4))*exp(-((z_1-z_2)*(z_1-
         z_2))/(4*D*t)));
      }
   '''
po = np.array([1e-4,100000])

def int_simp2d_c(N,po,t):
   total = weave.inline(code, ['N','po','t'], headers=['<math.h>'], support_code = support)
   return total

def test_func(p,t):
   data = []
   for i in t:
      data.append(int_simp2d_c(400,p,i))
```

```
    return data

start = time.time()
plsq = leastsq(errfunc, po, args = (x,t,weights))[0]
end = time.time()
print end - start
print plsq



fig = plt.figure(0, (9,6))
ax = fig.add_subplot(111)
ax.set_xscale('log')
ax.set_ylim(bottom=0, top=(max(x)+10))
ax.set_xlim(left=5e-7)
plt.scatter(t, x, color = 'red', marker = '^')
#plt.plot(t, test_func([plsq[0], po[1]],t), color = 'green', lw = 1)
plt.plot(t, test_func(plsq,t), color = 'blue', lw = 2)
plt.grid()
#plt.title('\t$A = %5.4f | V_x = %0.4f$' % (plsq[0], plsq[1]))
plt.show()
```

A.4 One-Focus Gaussian-Lorentzian Code with Triplet Fitting

```
import numpy as np
import pylab as plt
from scipy.optimize import leastsq
import scipy.weave as weave
import time
errfunc = lambda p, x, t, w: np.abs((test_func(p,t)-x)/np.sqrt(w))
f = open('c:/Users/jgermann/Desktop/Python/Data/1 focus 10 mmpersecond x motion seed 2.xls',
'r')
dump = []
for line in f:
dump.append(line.split('\t'))
sp = 6
dump = np.array(dump)
t = dump[sp:,0].astype('float')
x = dump[sp:,1].astype('float')
weights = dump[sp:,17].astype('float')
##weightfunc = np.zeros(len(weights))
##for i in range(len(x)):
## print i
## if x[i] > 1.5:
```

```
## weightfunc[i] = weights[i]
## bottomvalue = weights[i]
## else:
## weightfunc[i] = bottomvalue
code = \
r'''
double total = 0.;
double h_x = 2./(N-1);
double h_y = 2./(N-1);
double x; double y; double w;
int i; int j;
for(i=1;i<=N+1;i++)
{
for(j=1;j<=N+1;j++)
{
x = i*h_x;
y = j*h_y;
w = weight(i,j,N);
total += w*func(x,y,po[1],t, po[2], po[3], po[4]);
}
}
total *= 4./(9.*N*N);
total = 1+((po[0])*total/sqrt(t));
return_val = total;
'''
support = \
r'''
double weight( int i, int j, int N )
{
if(i==1 || i==N+1)
{
if(j==1 || j==N+1) return(1.);
if(j%2==0) return(4.);
else return(2.);
}
if(i%2==0)
{
if(j==N+1) return(4.);
if(j%2==0) return(16.);
else return(8.);
}
else
{
if(j==N+1) return(2.);
if(j%2==0) return(8.);
```

```
else return(4.);
}
}
double func( double z_1, double z_2, double v_x, double t, double tp1, double tp2, double tl )
{
double z_r = 0.98; double w_0 = 0.5; double D = 220;
double deltax = 0;
double wsqr_1 = (w_0*w_0)*(1+(z_1/z_r)*(z_1/z_r));
double wsqr_2 = (w_0*w_0)*(1+(z_2/z_r)*(z_2/z_r));
return((tp1+tp2*exp(-t/tl))*(1/(wsqr_1*wsqr_2))*(1/(2*D*t + wsqr_1/8 + wsqr_2/8))*exp(-
((v_x*t-deltax)*(v_x*t-deltax))/(4*D*t + wsqr_1/4 + wsqr_2/4))*exp(-((z_1-z_2)*(z_1-
z_2))/(4*D*t)));
}
'''
po = np.array([1e-2, 10000, 1e-3, 1e-3, 1])
def int_simp2d_c(N,po,t):
total = weave.inline(code, ['N','po','t'], headers=['<math.h>'], support_code = support)
return total
def test_func(p,t):
data = []
for i in t:
data.append(int_simp2d_c(400,p,i))
return data
start = time.time()
plsq = leastsq(errfunc, po, args = (x,t,weights))[0]
end = time.time()
print end - start
print plsq
fig = plt.figure(0, (9,6))
ax = fig.add_subplot(111)
ax.set_xscale('log')
ax.set_ylim(bottom=0, top=(max(x)+10))
ax.set_xlim(left=5e-7)
plt.scatter(t, x, color = 'red', marker = '^')
plt.plot(t, test_func(plsq,t), color = 'blue', lw = 2)
plt.grid()
plt.show()
```

A.5 Two-focus 3D Gaussian Code

```
import time
import numpy as np
from scipy.optimize import leastsq
import pylab as plt
```

```
#These are the constants for the foci and physical system
z_r = 0.98
w_0 = 0.3535534
D = 220
p0 = [51, 72, 20, 50, -500]
distx = 0.250
disty = 0
distz = 0
crosstalkmatrix = [[.854, .146],
                   [.146, .854]]

distarrayx = [0, distx, distx/2, distx/2, -distx, 0, -distx/2, -distx/2, -distx/2, distx/2, 0, 0, -distx/2,
        distx/2, 0, 0]
distarrayy = [0, 0, -disty/2, disty/2, 0, 0, -disty/2, disty/2, disty/2, disty/2, 0, disty, -disty/2, -
        disty/2, -disty, 0]
distarrayz = [0, 0, distz, distz, 0, 0, distz, distz, -distz, -distz, 0, 0, -distz, -distz, 0, 0]
cordata = []
a = []
b = []
c = []
d = []
ctmatrix = [[crosstalkmatrix[0][0]*crosstalkmatrix[0][0],
        crosstalkmatrix[0][0]*crosstalkmatrix[1][0], crosstalkmatrix[1][0]*crosstalkmatrix[0][0],
        crosstalkmatrix[1][0]*crosstalkmatrix[1][0]],
        [crosstalkmatrix[0][0]*crosstalkmatrix[0][1],
        crosstalkmatrix[0][0]*crosstalkmatrix[1][1], crosstalkmatrix[1][0]*crosstalkmatrix[0][1],
        crosstalkmatrix[1][0]*crosstalkmatrix[1][1]],
        [crosstalkmatrix[0][1]*crosstalkmatrix[0][0],
        crosstalkmatrix[0][1]*crosstalkmatrix[1][0], crosstalkmatrix[1][1]*crosstalkmatrix[0][0],
        crosstalkmatrix[1][1]*crosstalkmatrix[1][0]],
        [crosstalkmatrix[0][1]*crosstalkmatrix[0][1],
        crosstalkmatrix[0][1]*crosstalkmatrix[1][1], crosstalkmatrix[1][1]*crosstalkmatrix[0][1],
        crosstalkmatrix[1][1]*crosstalkmatrix[1][1]]]

func = lambda t, Tvx, Tvy, Tvz, Td, deltax, deltay, deltaz:  ((1+t/Td)**-
        1)*((np.sqrt(1+((w_0/z_r)**2)*t/Td))**-1)*np.exp(-((t/Tvx-
        deltax/w_0)**2)/(1+t/Td))*np.exp(-((t/Tvy-deltay/w_0)**2)/(1+t/Td))*np.exp(-((t/Tvz-
        deltaz/z_r)**2)/(1+((w_0/z_r)**2)*t/Td))


f_name = "c:/users/jgermann/desktop/python/data/half mmpers 500 nanometer separation
        random seed.xls"
f = open(f_name, 'r')
```

```
data = []

for line in f:
   data.append(line.split('\t'))

f.close()

data = np.array(data)

sp=6
td = data[sp:,0].astype(float)
ctdata = np.zeros(4*len(td))
autofunc = np.zeros(4*len(td))
weight=np.zeros(4*len(td))
for i in range(len(td)):              #Correlation arrays
   autofunc[i] = data[i+sp,1].astype('float')
   autofunc[1*len(td)+i] = data[i+sp,2].astype('float')
   autofunc[2*len(td)+i] = data[i+sp,5].astype('float')
   autofunc[3*len(td)+i] = data[i+sp,6].astype('float')
   weight[i] = data[i+sp,17].astype('float')
   weight[len(td)+i] = data[i+sp,18].astype('float')
   weight[2*len(td)+i] = data[i+sp,21].astype('float')
   weight[3*len(td)+i] = data[i+sp,22].astype('float')


def multifunc(p, t):
   cordata = []
   subdistarrayx = [0, distx, -distx, 0]
   for i in range(4):
      for j in t:
         cordata.append(1+p[i]*func(j, w_0/p[4], 1e21, 1e21, (w_0**2)/4*D, subdistarrayx[i], 0,
         0))
   a = cordata[0:1*len(td)]
   b = cordata[1*len(td):2*len(td)]
   c = cordata[2*len(td):3*len(td)]
   d = cordata[3*len(td):4*len(td)]
   for k in range(len(td)):    #Each time point
      for l in range(4): #Each correlation fucntion
         ctdata[l*len(td)+k] = ctmatrix[l][0]*a[k] + ctmatrix[l][1]*b[k] + ctmatrix[l][2]*c[k] +
         ctmatrix[l][3]*d[k] #Creates array of crosstalk model
   return ctdata


fitfunc = lambda p, xx: multifunc(p, xx)
errfunc = lambda p, xx, y, w: (fitfunc(p,xx)-y)/(np.sqrt(w))
```

```
start = time.time()
plsq = leastsq(errfunc, p0[:], args = (td, autofunc, weight))[0]
end = time.time()
q = multifunc(plsq, td)

print plsq

def plotsplan(i):       #Plotting function
    fig = plt.figure(0, (9,6))
    ax = fig.add_subplot(111)
    ax.set_xscale('log')
    ax.set_ylim(bottom=0, top=(max(q)+10))
    ax.set_xlim(left=5e-7)
    plt.scatter(td, autofunc[i*len(td):1*len(td)*(i+1)], color = 'red', marker = '^')
    plt.plot(td, q[i*len(td):len(td)*(i+1)], color = 'blue', lw = 2)
    plt.grid()
    plt.show()


print ((end-start)/60)

plotsplan(0)
plotsplan(1)
plotsplan(2)
plotsplan(3)
```

A.6 Two-focus Gaussian-Lorentzian Code

```
import numpy as np
import pylab as plt
from scipy.optimize import leastsq
import scipy.weave as weave
import time

errfunc = lambda p, x, t, w:  np.abs((test_func(p,t)-x)/np.sqrt(w))

f = open('c:/Users/jgermann/Desktop/Python/Data/1 focus 10 mmpersecond x motion seed 2.xls',
         'r')
dump = []
for line in f:
    dump.append(line.split('\t'))

sp = 6
dump = np.array(dump)
t = dump[sp:,0].astype('float')
```

```
x = dump[sp:,1].astype('float')
weights = dump[sp:,17].astype('float')

##weightfunc = np.zeros(len(weights))

##for i in range(len(x)):
##    print i
##    if x[i] > 1.5:
##        weightfunc[i] = weights[i]
##        bottomvalue = weights[i]
##    else:
##        weightfunc[i] = bottomvalue

code = \
   r'''
   double total = 0.;
   double h_x = 2./(N-1);
   double h_y = 2./(N-1);
   double x; double y; double w;
   int i; int j;


   for(i=1;i<=N+1;i++)
      {
      for(j=1;j<=N+1;j++)
         {
         x = i*h_x;
         y = j*h_y;
         w = weight(i,j,N);
         total += w*func(x,y,po[1],t, po[2], po[3], po[4]);
         }
      }
   total *= 4./(9.*N*N);
   total = 1+((po[0])*total/sqrt(t));
   return_val = total;
   '''

support = \
   r'''
   double weight( int i, int j, int N )
      {
      if(i==1 || i==N+1)
         {
         if(j==1 || j==N+1) return(1.);
         if(j%2==0) return(4.);
```

```
            else return(2.);
            }
        if(i%2==0)
            {
            if(j==N+1) return(4.);
            if(j%2==0) return(16.);
            else return(8.);
            }
        else
            {
            if(j==N+1) return(2.);
            if(j%2==0) return(8.);
            else return(4.);
            }
        }

    double func( double z_1, double z_2, double v_x, double t, double tp1, double tp2, double tl )
        {
        double z_r = 0.98; double w_0 = 0.5; double D = 220;
        double deltax = 0;
        double wsqr_1 = (w_0*w_0)*(1+(z_1/z_r)*(z_1/z_r));
        double wsqr_2 = (w_0*w_0)*(1+(z_2/z_r)*(z_2/z_r));
        return((tp1+tp2*exp(-t/tl))*(1/(wsqr_1*wsqr_2))*(1/(2*D*t + wsqr_1/8 + wsqr_2/8))*exp(-
            ((v_x*t-deltax)*(v_x*t-deltax))/(4*D*t + wsqr_1/4 + wsqr_2/4))*exp(-((z_1-z_2)*(z_1-
            z_2))/(4*D*t)));
        }
    '''
po = np.array([1e-2, 10000, 1e-3, 1e-3, 1])

def int_simp2d_c(N,po,t):
    total = weave.inline(code, ['N','po','t'], headers=['<math.h>'], support_code = support)
    return total

def test_func(p,t):
    data = []
    for i in t:
        data.append(int_simp2d_c(400,p,i))
    return data

start = time.time()
plsq = leastsq(errfunc, po, args = (x,t,weights))[0]
end = time.time()
print end - start
print plsq
```

```
fig = plt.figure(0, (9,6))
ax = fig.add_subplot(111)
ax.set_xscale('log')
ax.set_ylim(bottom=0, top=(max(x)+10))
ax.set_xlim(left=5e-7)
plt.scatter(t, x, color = 'red', marker = '^')
plt.plot(t, test_func(plsq,t), color = 'blue', lw = 2)
plt.grid()
plt.show()
```

A.7 Four-focus 3D Gaussian Code

```
import time
import numpy as np
from scipy.optimize import leastsq
import pylab as plt

#These are the constants for the foci and physical system
z_r = 0.98
w_0 = 0.5
D = 220
p0 = [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, -10000,
10000, -10000]
distx = 0.5
disty = 0.5
distz = 0.98
crosstalkmatrix = [[.84, .134, .021, .003],
                   [.003, .84, .134, .021],
                   [.021, .003, .84, .134],
                   [.134, .021, .003, .84]]   #Crosstalk matrix, decimal style

distarrayx = [0, distx, distx/2, distx/2, -distx, 0, -distx/2, -distx/2, -distx/2, distx/2, 0, 0, -distx/2,
distx/2, 0, 0]
distarrayy = [0, 0, -disty/2, disty/2, 0, 0, -disty/2, disty/2, disty/2, disty/2, 0, disty, -disty/2, -
disty/2, -disty, 0]
distarrayz = [0, 0, distz, distz, 0, 0, distz, distz, -distz, -distz, 0, 0, -distz, -distz, 0, 0]
cordata = []
aa = []  #arrays for storing fitted data before crosstalk takes effect
ab = []
ac = []
ad = []
ba = []
bb = []
bc = []
bd = []
```

```
ca = []
cb = []
cc = []
cd = []
da = []
db = []
dc = []
dd = []
ctmatrix = [[crosstalkmatrix[0][0]*crosstalkmatrix[0][0],
crosstalkmatrix[0][0]*crosstalkmatrix[1][0], crosstalkmatrix[0][0]*crosstalkmatrix[2][0],
crosstalkmatrix[0][0]*crosstalkmatrix[3][0], crosstalkmatrix[1][0]*crosstalkmatrix[0][0],
crosstalkmatrix[1][0]*crosstalkmatrix[1][0], crosstalkmatrix[1][0]*crosstalkmatrix[2][0],
crosstalkmatrix[1][0]*crosstalkmatrix[3][0], crosstalkmatrix[2][0]*crosstalkmatrix[0][0],
crosstalkmatrix[2][0]*crosstalkmatrix[1][0], crosstalkmatrix[2][0]*crosstalkmatrix[2][0],
crosstalkmatrix[2][0]*crosstalkmatrix[3][0], crosstalkmatrix[3][0]*crosstalkmatrix[0][0],
crosstalkmatrix[3][0]*crosstalkmatrix[1][0], crosstalkmatrix[3][0]*crosstalkmatrix[2][0],
crosstalkmatrix[3][0]*crosstalkmatrix[3][0]],
    [crosstalkmatrix[0][0]*crosstalkmatrix[0][1], crosstalkmatrix[0][0]*crosstalkmatrix[1][1],
crosstalkmatrix[0][0]*crosstalkmatrix[2][1], crosstalkmatrix[0][0]*crosstalkmatrix[3][1],
crosstalkmatrix[1][0]*crosstalkmatrix[0][1], crosstalkmatrix[1][0]*crosstalkmatrix[1][1],
crosstalkmatrix[1][0]*crosstalkmatrix[2][1], crosstalkmatrix[1][0]*crosstalkmatrix[3][1],
crosstalkmatrix[2][0]*crosstalkmatrix[0][1], crosstalkmatrix[2][0]*crosstalkmatrix[1][1],
crosstalkmatrix[2][0]*crosstalkmatrix[2][1], crosstalkmatrix[2][0]*crosstalkmatrix[3][1],
crosstalkmatrix[3][0]*crosstalkmatrix[0][1], crosstalkmatrix[3][0]*crosstalkmatrix[1][1],
crosstalkmatrix[3][0]*crosstalkmatrix[2][1], crosstalkmatrix[3][0]*crosstalkmatrix[3][1]],
    [crosstalkmatrix[0][0]*crosstalkmatrix[0][2], crosstalkmatrix[0][0]*crosstalkmatrix[1][2],
crosstalkmatrix[0][0]*crosstalkmatrix[2][2], crosstalkmatrix[0][0]*crosstalkmatrix[3][2],
crosstalkmatrix[1][0]*crosstalkmatrix[0][2], crosstalkmatrix[1][0]*crosstalkmatrix[1][2],
crosstalkmatrix[1][0]*crosstalkmatrix[2][2], crosstalkmatrix[1][0]*crosstalkmatrix[3][2],
crosstalkmatrix[2][0]*crosstalkmatrix[0][2], crosstalkmatrix[2][0]*crosstalkmatrix[1][2],
crosstalkmatrix[2][0]*crosstalkmatrix[2][2], crosstalkmatrix[2][0]*crosstalkmatrix[3][2],
crosstalkmatrix[3][0]*crosstalkmatrix[0][2], crosstalkmatrix[3][0]*crosstalkmatrix[1][2],
crosstalkmatrix[3][0]*crosstalkmatrix[2][2], crosstalkmatrix[3][0]*crosstalkmatrix[3][2]],
    [crosstalkmatrix[0][0]*crosstalkmatrix[0][3], crosstalkmatrix[0][0]*crosstalkmatrix[1][3],
crosstalkmatrix[0][0]*crosstalkmatrix[2][3], crosstalkmatrix[0][0]*crosstalkmatrix[3][3],
crosstalkmatrix[1][0]*crosstalkmatrix[0][3], crosstalkmatrix[1][0]*crosstalkmatrix[1][3],
crosstalkmatrix[1][0]*crosstalkmatrix[2][3], crosstalkmatrix[1][0]*crosstalkmatrix[3][3],
crosstalkmatrix[2][0]*crosstalkmatrix[0][3], crosstalkmatrix[2][0]*crosstalkmatrix[1][3],
crosstalkmatrix[2][0]*crosstalkmatrix[2][3], crosstalkmatrix[2][0]*crosstalkmatrix[3][3],
crosstalkmatrix[3][0]*crosstalkmatrix[0][3], crosstalkmatrix[3][0]*crosstalkmatrix[1][3],
crosstalkmatrix[3][0]*crosstalkmatrix[2][3], crosstalkmatrix[3][0]*crosstalkmatrix[3][3]],
    [crosstalkmatrix[0][1]*crosstalkmatrix[0][0], crosstalkmatrix[0][1]*crosstalkmatrix[1][0],
crosstalkmatrix[0][1]*crosstalkmatrix[2][0], crosstalkmatrix[0][1]*crosstalkmatrix[3][0],
crosstalkmatrix[1][1]*crosstalkmatrix[0][0], crosstalkmatrix[1][1]*crosstalkmatrix[1][0],
crosstalkmatrix[1][1]*crosstalkmatrix[2][0], crosstalkmatrix[1][1]*crosstalkmatrix[3][0],
```

crosstalkmatrix[2][1]*crosstalkmatrix[0][0], crosstalkmatrix[2][1]*crosstalkmatrix[1][0],
crosstalkmatrix[2][1]*crosstalkmatrix[2][0], crosstalkmatrix[2][1]*crosstalkmatrix[3][0],
crosstalkmatrix[3][1]*crosstalkmatrix[0][0], crosstalkmatrix[3][1]*crosstalkmatrix[1][0],
crosstalkmatrix[3][1]*crosstalkmatrix[2][0], crosstalkmatrix[3][1]*crosstalkmatrix[3][0]],
[crosstalkmatrix[0][1]*crosstalkmatrix[0][1], crosstalkmatrix[0][1]*crosstalkmatrix[1][1],
crosstalkmatrix[0][1]*crosstalkmatrix[2][1], crosstalkmatrix[0][1]*crosstalkmatrix[3][1],
crosstalkmatrix[1][1]*crosstalkmatrix[0][1], crosstalkmatrix[1][1]*crosstalkmatrix[1][1],
crosstalkmatrix[1][1]*crosstalkmatrix[2][1], crosstalkmatrix[1][1]*crosstalkmatrix[3][1],
crosstalkmatrix[2][1]*crosstalkmatrix[0][1], crosstalkmatrix[2][1]*crosstalkmatrix[1][1],
crosstalkmatrix[2][1]*crosstalkmatrix[2][1], crosstalkmatrix[2][1]*crosstalkmatrix[3][1],
crosstalkmatrix[3][1]*crosstalkmatrix[0][1], crosstalkmatrix[3][1]*crosstalkmatrix[1][1],
crosstalkmatrix[3][1]*crosstalkmatrix[2][1], crosstalkmatrix[3][1]*crosstalkmatrix[3][1]],
[crosstalkmatrix[0][1]*crosstalkmatrix[0][2], crosstalkmatrix[0][1]*crosstalkmatrix[1][2],
crosstalkmatrix[0][1]*crosstalkmatrix[2][2], crosstalkmatrix[0][1]*crosstalkmatrix[3][2],
crosstalkmatrix[1][1]*crosstalkmatrix[0][2], crosstalkmatrix[1][1]*crosstalkmatrix[1][2],
crosstalkmatrix[1][1]*crosstalkmatrix[2][2], crosstalkmatrix[1][1]*crosstalkmatrix[3][2],
crosstalkmatrix[2][1]*crosstalkmatrix[0][2], crosstalkmatrix[2][1]*crosstalkmatrix[1][2],
crosstalkmatrix[2][1]*crosstalkmatrix[2][2], crosstalkmatrix[2][1]*crosstalkmatrix[3][2],
crosstalkmatrix[3][1]*crosstalkmatrix[0][2], crosstalkmatrix[3][1]*crosstalkmatrix[1][2],
crosstalkmatrix[3][1]*crosstalkmatrix[2][2], crosstalkmatrix[3][1]*crosstalkmatrix[3][2]],
[crosstalkmatrix[0][1]*crosstalkmatrix[0][3], crosstalkmatrix[0][1]*crosstalkmatrix[1][3],
crosstalkmatrix[0][1]*crosstalkmatrix[2][3], crosstalkmatrix[0][1]*crosstalkmatrix[3][3],
crosstalkmatrix[1][1]*crosstalkmatrix[0][3], crosstalkmatrix[1][1]*crosstalkmatrix[1][3],
crosstalkmatrix[1][1]*crosstalkmatrix[2][3], crosstalkmatrix[1][1]*crosstalkmatrix[3][3],
crosstalkmatrix[2][1]*crosstalkmatrix[0][3], crosstalkmatrix[2][1]*crosstalkmatrix[1][3],
crosstalkmatrix[2][1]*crosstalkmatrix[2][3], crosstalkmatrix[2][1]*crosstalkmatrix[3][3],
crosstalkmatrix[3][1]*crosstalkmatrix[0][3], crosstalkmatrix[3][1]*crosstalkmatrix[1][3],
crosstalkmatrix[3][1]*crosstalkmatrix[2][3], crosstalkmatrix[3][1]*crosstalkmatrix[3][3]],
[crosstalkmatrix[0][2]*crosstalkmatrix[0][0], crosstalkmatrix[0][2]*crosstalkmatrix[1][0],
crosstalkmatrix[0][2]*crosstalkmatrix[2][0], crosstalkmatrix[0][2]*crosstalkmatrix[3][0],
crosstalkmatrix[1][2]*crosstalkmatrix[0][0], crosstalkmatrix[1][2]*crosstalkmatrix[1][0],
crosstalkmatrix[1][2]*crosstalkmatrix[2][0], crosstalkmatrix[1][2]*crosstalkmatrix[3][0],
crosstalkmatrix[2][2]*crosstalkmatrix[0][0], crosstalkmatrix[2][2]*crosstalkmatrix[1][0],
crosstalkmatrix[2][2]*crosstalkmatrix[2][0], crosstalkmatrix[2][2]*crosstalkmatrix[3][0],
crosstalkmatrix[3][2]*crosstalkmatrix[0][0], crosstalkmatrix[3][2]*crosstalkmatrix[1][0],
crosstalkmatrix[3][2]*crosstalkmatrix[2][0], crosstalkmatrix[3][2]*crosstalkmatrix[3][0]],
[crosstalkmatrix[0][2]*crosstalkmatrix[0][1], crosstalkmatrix[0][2]*crosstalkmatrix[1][1],
crosstalkmatrix[0][2]*crosstalkmatrix[2][1], crosstalkmatrix[0][2]*crosstalkmatrix[3][1],
crosstalkmatrix[1][2]*crosstalkmatrix[0][1], crosstalkmatrix[1][2]*crosstalkmatrix[1][1],
crosstalkmatrix[1][2]*crosstalkmatrix[2][1], crosstalkmatrix[1][2]*crosstalkmatrix[3][1],
crosstalkmatrix[2][2]*crosstalkmatrix[0][1], crosstalkmatrix[2][2]*crosstalkmatrix[1][1],
crosstalkmatrix[2][2]*crosstalkmatrix[2][1], crosstalkmatrix[2][2]*crosstalkmatrix[3][1],
crosstalkmatrix[3][2]*crosstalkmatrix[0][1], crosstalkmatrix[3][2]*crosstalkmatrix[1][1],
crosstalkmatrix[3][2]*crosstalkmatrix[2][1], crosstalkmatrix[3][2]*crosstalkmatrix[3][1]],

[crosstalkmatrix[0][2]*crosstalkmatrix[0][2], crosstalkmatrix[0][2]*crosstalkmatrix[1][2], crosstalkmatrix[0][2]*crosstalkmatrix[2][2], crosstalkmatrix[0][2]*crosstalkmatrix[3][2], crosstalkmatrix[1][2]*crosstalkmatrix[0][2], crosstalkmatrix[1][2]*crosstalkmatrix[1][2], crosstalkmatrix[1][2]*crosstalkmatrix[2][2], crosstalkmatrix[1][2]*crosstalkmatrix[3][2], crosstalkmatrix[2][2]*crosstalkmatrix[0][2], crosstalkmatrix[2][2]*crosstalkmatrix[1][2], crosstalkmatrix[2][2]*crosstalkmatrix[2][2], crosstalkmatrix[2][2]*crosstalkmatrix[3][2], crosstalkmatrix[3][2]*crosstalkmatrix[0][2], crosstalkmatrix[3][2]*crosstalkmatrix[1][2], crosstalkmatrix[3][2]*crosstalkmatrix[2][2], crosstalkmatrix[3][2]*crosstalkmatrix[3][2]],
[crosstalkmatrix[0][2]*crosstalkmatrix[0][3], crosstalkmatrix[0][2]*crosstalkmatrix[1][3], crosstalkmatrix[0][2]*crosstalkmatrix[2][3], crosstalkmatrix[0][2]*crosstalkmatrix[3][3], crosstalkmatrix[1][2]*crosstalkmatrix[0][3], crosstalkmatrix[1][2]*crosstalkmatrix[1][3], crosstalkmatrix[1][2]*crosstalkmatrix[2][3], crosstalkmatrix[1][2]*crosstalkmatrix[3][3], crosstalkmatrix[2][2]*crosstalkmatrix[0][3], crosstalkmatrix[2][2]*crosstalkmatrix[1][3], crosstalkmatrix[2][2]*crosstalkmatrix[2][3], crosstalkmatrix[2][2]*crosstalkmatrix[3][3], crosstalkmatrix[3][2]*crosstalkmatrix[0][3], crosstalkmatrix[3][2]*crosstalkmatrix[1][3], crosstalkmatrix[3][2]*crosstalkmatrix[2][3], crosstalkmatrix[3][2]*crosstalkmatrix[3][3]],
[crosstalkmatrix[0][3]*crosstalkmatrix[0][0], crosstalkmatrix[0][3]*crosstalkmatrix[1][0], crosstalkmatrix[0][3]*crosstalkmatrix[2][0], crosstalkmatrix[0][3]*crosstalkmatrix[3][0], crosstalkmatrix[1][3]*crosstalkmatrix[0][0], crosstalkmatrix[1][3]*crosstalkmatrix[1][0], crosstalkmatrix[1][3]*crosstalkmatrix[2][0], crosstalkmatrix[1][3]*crosstalkmatrix[3][0], crosstalkmatrix[2][3]*crosstalkmatrix[0][0], crosstalkmatrix[2][3]*crosstalkmatrix[1][0], crosstalkmatrix[2][3]*crosstalkmatrix[2][0], crosstalkmatrix[2][3]*crosstalkmatrix[3][0], crosstalkmatrix[3][3]*crosstalkmatrix[0][0], crosstalkmatrix[3][3]*crosstalkmatrix[1][0], crosstalkmatrix[3][3]*crosstalkmatrix[2][0], crosstalkmatrix[3][3]*crosstalkmatrix[3][0]],
[crosstalkmatrix[0][3]*crosstalkmatrix[0][1], crosstalkmatrix[0][3]*crosstalkmatrix[1][1], crosstalkmatrix[0][3]*crosstalkmatrix[2][1], crosstalkmatrix[0][3]*crosstalkmatrix[3][1], crosstalkmatrix[1][3]*crosstalkmatrix[0][1], crosstalkmatrix[1][3]*crosstalkmatrix[1][1], crosstalkmatrix[1][3]*crosstalkmatrix[2][1], crosstalkmatrix[1][3]*crosstalkmatrix[3][1], crosstalkmatrix[2][3]*crosstalkmatrix[0][1], crosstalkmatrix[2][3]*crosstalkmatrix[1][1], crosstalkmatrix[2][3]*crosstalkmatrix[2][1], crosstalkmatrix[2][3]*crosstalkmatrix[3][1], crosstalkmatrix[3][3]*crosstalkmatrix[0][1], crosstalkmatrix[3][3]*crosstalkmatrix[1][1], crosstalkmatrix[3][3]*crosstalkmatrix[2][1], crosstalkmatrix[3][3]*crosstalkmatrix[3][1]],
[crosstalkmatrix[0][3]*crosstalkmatrix[0][2], crosstalkmatrix[0][3]*crosstalkmatrix[1][2], crosstalkmatrix[0][3]*crosstalkmatrix[2][2], crosstalkmatrix[0][3]*crosstalkmatrix[3][2], crosstalkmatrix[1][3]*crosstalkmatrix[0][2], crosstalkmatrix[1][3]*crosstalkmatrix[1][2], crosstalkmatrix[1][3]*crosstalkmatrix[2][2], crosstalkmatrix[1][3]*crosstalkmatrix[3][2], crosstalkmatrix[2][3]*crosstalkmatrix[0][2], crosstalkmatrix[2][3]*crosstalkmatrix[1][2], crosstalkmatrix[2][3]*crosstalkmatrix[2][2], crosstalkmatrix[2][3]*crosstalkmatrix[3][2], crosstalkmatrix[3][3]*crosstalkmatrix[0][2], crosstalkmatrix[3][3]*crosstalkmatrix[1][2], crosstalkmatrix[3][3]*crosstalkmatrix[2][2], crosstalkmatrix[3][3]*crosstalkmatrix[3][2]],
[crosstalkmatrix[0][3]*crosstalkmatrix[0][3], crosstalkmatrix[0][3]*crosstalkmatrix[1][3], crosstalkmatrix[0][3]*crosstalkmatrix[2][3], crosstalkmatrix[0][3]*crosstalkmatrix[3][3], crosstalkmatrix[1][3]*crosstalkmatrix[0][3], crosstalkmatrix[1][3]*crosstalkmatrix[1][3], crosstalkmatrix[1][3]*crosstalkmatrix[2][3], crosstalkmatrix[1][3]*crosstalkmatrix[3][3], crosstalkmatrix[2][3]*crosstalkmatrix[0][3], crosstalkmatrix[2][3]*crosstalkmatrix[1][3],

```
crosstalkmatrix[2][3]*crosstalkmatrix[2][3], crosstalkmatrix[2][3]*crosstalkmatrix[3][3],
crosstalkmatrix[3][3]*crosstalkmatrix[0][3], crosstalkmatrix[3][3]*crosstalkmatrix[1][3],
crosstalkmatrix[3][3]*crosstalkmatrix[2][3], crosstalkmatrix[3][3]*crosstalkmatrix[3][3]]]

func = lambda t, Tvx, Tvy, Tvz, Td, deltax, deltay, deltaz:  ((1+t/Td)**-
1)*((np.sqrt(1+((w_0/z_r)**2)*t/Td))**-1)*np.exp(-((t/Tvx-deltax/w_0)**2)/(1+t/Td))*np.exp(-
((t/Tvy-deltay/w_0)**2)/(1+t/Td))*np.exp(-((t/Tvz-deltaz/z_r)**2)/(1+((w_0/z_r)**2)*t/Td))


f_name = "c:/users/jgermann/desktop/python/data/Oct2012runs/4focusnotripx10y10z10.xls"
f = open(f_name, 'r')

data = []

for line in f:
   data.append(line.split('\t'))

f.close()

data = np.array(data)

sp=6
td = data[sp:,0].astype(float)
autofunc = np.zeros(16*len(td))
ctdata = np.zeros(16*len(td))
weights = np.zeros(16*len(td))
for i in range(len(td)):              #Correlation arrays
   autofunc[i] = data[sp+i,1].astype('float')
   autofunc[1*len(td)+i] = data[sp+i,2].astype('float')
   autofunc[2*len(td)+i] = data[sp+i,3].astype('float')
   autofunc[3*len(td)+i] = data[sp+i,4].astype('float')
   autofunc[4*len(td)+i] = data[sp+i,5].astype('float')
   autofunc[5*len(td)+i] = data[sp+i,6].astype('float')
   autofunc[6*len(td)+i] = data[sp+i,7].astype('float')
   autofunc[7*len(td)+i] = data[sp+i,8].astype('float')
   autofunc[8*len(td)+i] = data[sp+i,9].astype('float')
   autofunc[9*len(td)+i] = data[sp+i,10].astype('float')
   autofunc[10*len(td)+i] = data[sp+i,11].astype('float')
   autofunc[11*len(td)+i] = data[sp+i,12].astype('float')
   autofunc[12*len(td)+i] = data[sp+i,13].astype('float')
   autofunc[13*len(td)+i] = data[sp+i,14].astype('float')
   autofunc[14*len(td)+i] = data[sp+i,15].astype('float')
   autofunc[15*len(td)+i] = data[sp+i,16].astype('float')
   weights[i] = data[sp+i,17].astype('float')
   weights[1*len(td)+i] = data[sp+i,18].astype('float')
```

```
    weights[2*len(td)+i] = data[sp+i,19].astype('float')
    weights[3*len(td)+i] = data[sp+i,20].astype('float')
    weights[4*len(td)+i] = data[sp+i,21].astype('float')
    weights[5*len(td)+i] = data[sp+i,22].astype('float')
    weights[6*len(td)+i] = data[sp+i,23].astype('float')
    weights[7*len(td)+i] = data[sp+i,24].astype('float')
    weights[8*len(td)+i] = data[sp+i,25].astype('float')
    weights[9*len(td)+i] = data[sp+i,26].astype('float')
    weights[10*len(td)+i] = data[sp+i,27].astype('float')
    weights[11*len(td)+i] = data[sp+i,28].astype('float')
    weights[12*len(td)+i] = data[sp+i,29].astype('float')
    weights[13*len(td)+i] = data[sp+i,30].astype('float')
    weights[14*len(td)+i] = data[sp+i,31].astype('float')
    weights[15*len(td)+i] = data[sp+i,32].astype('float')

def multifunc(p, t):
    cordata = []
    for i in range(16):
        for j in t:
            cordata.append(1+p[i]*func(j, w_0/p[16], w_0/p[17], z_r/p[18], (w_0**2)/(8*D),
distarrayx[i], distarrayy[i], distarrayz[i]))
    aa = cordata[0:len(t)]
    ab = cordata[1*len(t):2*len(t)]
    ac = cordata[2*len(t):3*len(t)]
    ad = cordata[3*len(t):4*len(t)]
    ba = cordata[4*len(t):5*len(t)]
    bb = cordata[5*len(t):6*len(t)]
    bc = cordata[6*len(t):7*len(t)]
    bd = cordata[7*len(t):8*len(t)]
    ca = cordata[8*len(t):9*len(t)]
    cb = cordata[9*len(t):10*len(t)]
    cc = cordata[10*len(t):11*len(t)]
    cd = cordata[11*len(t):12*len(t)]
    da = cordata[12*len(t):13*len(t)]
    db = cordata[13*len(t):14*len(t)]
    dc = cordata[14*len(t):15*len(t)]
    dd = cordata[15*len(t):16*len(t)]
    for k in range(len(t)):    #Each time point
        for l in range(16): #Each correlation fucntion
            ctdata[l*len(td)+k] = ctmatrix[l][0]*aa[k] + ctmatrix[l][1]*ab[k] + ctmatrix[l][2]*ac[k] +
ctmatrix[l][3]*ad[k] + ctmatrix[l][4]*ba[k] + ctmatrix[l][5]*bb[k] + ctmatrix[l][6]*bc[k] +
ctmatrix[l][7]*bd[k] + ctmatrix[l][8]*ca[k] + ctmatrix[l][9]*cb[k] + ctmatrix[l][10]*cc[k] +
ctmatrix[l][11]*cd[k] + ctmatrix[l][12]*da[k] + ctmatrix[l][13]*db[k] + ctmatrix[l][14]*dc[k] +
ctmatrix[l][15]*dd[k] #Creates array of crosstalk model
    return ctdata
```

```
fitfunc = lambda p, xx: multifunc(p, xx)
errfunc = lambda p, xx, y, w: ((fitfunc(p,xx)-y)/np.sqrt(w))

start = time.time()
plsq = leastsq(errfunc, p0[:], args = (td, autofunc, weights))
end = time.time()
q = multifunc(plsq[0], td)

print plsq



def plotsplan(i):       #Plotting function
        fig = plt.figure(i, (9,6))
        ax = fig.add_subplot(111)
        ax.set_xscale('log')
        plt.scatter(td, autofunc[i*len(td):len(td)*(i+1)], color = 'red', marker = '^')
        plt.plot(td, q[i*len(td):len(td)*(i+1)], color = 'blue', lw = 2)
        plt.grid()
        plt.show()


print ((end-start)/60)


for i in range(16):
   plotsplan(i)
```

A.8 Four-Focus Gaussian-Lorentzian Code

```
import numpy as np
import pylab as plt
from scipy.optimize import leastsq

import time
import scipy.weave as weave


v = [0, 0, 0]  #vx, vy, vz in microns per second
v = np.array(v)
po = [0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, -
10000, -10000, -20000]
po = np.array(po)
```

```
cmo = [.8565, .1229, .0176, .0025]
crosstalkmatrix = [[cmo[0], cmo[1], cmo[2], cmo[3]],
                   [cmo[3], cmo[0], cmo[1], cmo[2]],
                   [cmo[2], cmo[3], cmo[0], cmo[1]],
                   [cmo[1], cmo[2], cmo[3], cmo[0]]]   #Crosstalk matrix, decimal style
distx = 0.5  #Measured or assigned distances along the x-axis
disty = 0.5  #Measured or assigned distances along the y-axis
distz = 0.98 #Measured or assigned distances along the z-axis
data = []  #Data is the measured correlation data
##ctma = []  #CrossTalk Matrix Arrays
aa = []  #arrays for storing fitted data before crosstalk takes effect
ab = []
ac = []
ad = []
ba = []
bb = []
bc = []
bd = []
ca = []
cb = []
cc = []
cd = []
da = []
db = []
dc = []
dd = []

distarrayx = np.array([0, distx, distx/2, distx/2, -distx, -distx/2, -distx/2, 0, -distx/2, distx/2, 0, 0, -
distx/2, distx/2, 0, 0])
distarrayy = np.array([0, 0, -disty/2, disty/2, 0, 0, -disty/2, disty/2, disty/2, disty/2, 0, disty, -
disty/2, -disty/2, -disty, 0])
distarrayz = np.array([0, 0, distz, distz, 0, 0, distz, distz, -distz, -distz, 0, 0, -distz, -distz, 0, 0])



ctma = np.zeros((16, 16))

for h in range(4):
    for g in range(4):
        ctma[4*h+g][0] = crosstalkmatrix[0][h]*crosstalkmatrix[0][g]
        ctma[4*h+g][1] = crosstalkmatrix[0][h]*crosstalkmatrix[1][g]
        ctma[4*h+g][2] = crosstalkmatrix[0][h]*crosstalkmatrix[2][g]
        ctma[4*h+g][3] = crosstalkmatrix[0][h]*crosstalkmatrix[3][g]
        ctma[4*h+g][4] = crosstalkmatrix[1][h]*crosstalkmatrix[0][g]
        ctma[4*h+g][5] = crosstalkmatrix[1][h]*crosstalkmatrix[1][g]
```

```
        ctma[4*h+g][6] = crosstalkmatrix[1][h]*crosstalkmatrix[2][g]
        ctma[4*h+g][7] = crosstalkmatrix[1][h]*crosstalkmatrix[3][g]
        ctma[4*h+g][8] = crosstalkmatrix[2][h]*crosstalkmatrix[0][g]
        ctma[4*h+g][9] = crosstalkmatrix[2][h]*crosstalkmatrix[1][g]
        ctma[4*h+g][10] = crosstalkmatrix[2][h]*crosstalkmatrix[2][g]
        ctma[4*h+g][11] = crosstalkmatrix[2][h]*crosstalkmatrix[3][g]
        ctma[4*h+g][12] = crosstalkmatrix[3][h]*crosstalkmatrix[0][g]
        ctma[4*h+g][13] = crosstalkmatrix[3][h]*crosstalkmatrix[1][g]
        ctma[4*h+g][14] = crosstalkmatrix[3][h]*crosstalkmatrix[2][g]
        ctma[4*h+g][15] = crosstalkmatrix[3][h]*crosstalkmatrix[3][g]


errfunc = lambda p, x, t, w:  np.abs((test_func(p,t)-x)/w) #Error function - to be minimized by
the leastsq module

f = open('c:/users/jgermann/desktop/python/data/4 foci no flow random seed.xls', 'r')  #File to
read data from
dump = []  #Data read
for line in f:
    dump.append(line.split('\t'))  #Extracting data

sp = 6    ##This is the starting point of the array, to cut out the afterpulsing
dump = np.array(dump)   #Convert list into array
t = dump[sp:,0].astype('float')    #Time array
x=np.zeros(16*len(t))
w=np.zeros(16*len(t))
for i in range(len(t)):              #Correlation arrays
    x[i] = dump[sp+i,1].astype('float')
    x[1*len(t)+i] = dump[sp+i,2].astype('float')
    x[2*len(t)+i] = dump[sp+i,3].astype('float')
    x[3*len(t)+i] = dump[sp+i,4].astype('float')
    x[4*len(t)+i] = dump[sp+i,5].astype('float')
    x[5*len(t)+i] = dump[sp+i,6].astype('float')
    x[6*len(t)+i] = dump[sp+i,7].astype('float')
    x[7*len(t)+i] = dump[sp+i,8].astype('float')
    x[8*len(t)+i] = dump[sp+i,9].astype('float')
    x[9*len(t)+i] = dump[sp+i,10].astype('float')
    x[10*len(t)+i] = dump[sp+i,11].astype('float')
    x[11*len(t)+i] = dump[sp+i,12].astype('float')
    x[12*len(t)+i] = dump[sp+i,13].astype('float')
    x[13*len(t)+i] = dump[sp+i,14].astype('float')
    x[14*len(t)+i] = dump[sp+i,15].astype('float')
    x[15*len(t)+i] = dump[sp+i,16].astype('float')
    w[i] = np.sqrt(dump[sp+i,17].astype('float'))
    w[1*len(t)+i] = np.sqrt(dump[sp+i,18].astype('float'))
```

```
    w[2*len(t)+i] = np.sqrt(dump[sp+i,19].astype('float'))
    w[3*len(t)+i] = np.sqrt(dump[sp+i,20].astype('float'))
    w[4*len(t)+i] = np.sqrt(dump[sp+i,21].astype('float'))
    w[5*len(t)+i] = np.sqrt(dump[sp+i,22].astype('float'))
    w[6*len(t)+i] = np.sqrt(dump[sp+i,23].astype('float'))
    w[7*len(t)+i] = np.sqrt(dump[sp+i,24].astype('float'))
    w[8*len(t)+i] = np.sqrt(dump[sp+i,25].astype('float'))
    w[9*len(t)+i] = np.sqrt(dump[sp+i,26].astype('float'))
    w[10*len(t)+i] = np.sqrt(dump[sp+i,27].astype('float'))
    w[11*len(t)+i] = np.sqrt(dump[sp+i,28].astype('float'))
    w[12*len(t)+i] = np.sqrt(dump[sp+i,29].astype('float'))
    w[13*len(t)+i] = np.sqrt(dump[sp+i,30].astype('float'))
    w[14*len(t)+i] = np.sqrt(dump[sp+i,31].astype('float'))
    w[15*len(t)+i] = np.sqrt(dump[sp+i,32].astype('float'))


ctdata = np.zeros(16*len(t))  #Ctdata will be the fitted data with crosstalk included

def test_func(p,t):     #integration of model done here
    data = []
    for j in range(16):
        for i in t:
            data.append(int_simp2d_c(400, p[j], p[16], p[17], p[18], distarrayx[j], distarrayy[j],
distarrayz[j], i))
    aa = data[0:len(t)]
    ab = data[1*len(t):2*len(t)]
    ac = data[2*len(t):3*len(t)]
    ad = data[3*len(t):4*len(t)]
    ba = data[4*len(t):5*len(t)]
    bb = data[5*len(t):6*len(t)]
    bc = data[6*len(t):7*len(t)]
    bd = data[7*len(t):8*len(t)]
    ca = data[8*len(t):9*len(t)]
    cb = data[9*len(t):10*len(t)]
    cc = data[10*len(t):11*len(t)]
    cd = data[11*len(t):12*len(t)]
    da = data[12*len(t):13*len(t)]
    db = data[13*len(t):14*len(t)]
    dc = data[14*len(t):15*len(t)]
    dd = data[15*len(t):16*len(t)]
    for k in range(len(t)):    #Each time point
        for l in range(16): #Each correlation fucntion
            ctdata[l*len(t)+k] = ctma[l][0]*aa[k] + ctma[l][1]*ab[k] + ctma[l][2]*ac[k] +
ctma[l][3]*ad[k] + ctma[l][4]*ba[k] + ctma[l][5]*bb[k] + ctma[l][6]*bc[k] + ctma[l][7]*bd[k] +
```

```
ctma[l][8]*ca[k] + ctma[l][9]*cb[k] + ctma[l][10]*cc[k] + ctma[l][11]*cd[k] + ctma[l][12]*da[k]
+ ctma[l][13]*db[k] + ctma[l][14]*dc[k] + ctma[l][15]*dd[k]  #Creates array of crosstalk model
    return ctdata         ##    return data


code = \
    r'''
    double total = 0.;
    double h_x = 4./(N-1);
    double h_y = 4./(N-1);
    double x; double y; double w;
    int i; int j;
    double amplitude = amp;

    for(i=1;i<=N+1;i++)
        {
        for(j=1;j<=N+1;j++)
            {
            x = i*h_x;
            y = j*h_y;
            w = weight(i,j,N);
            total += w*func(x, y, velx, vely, velz, dx, dy, dz, t);
            }
        }
    total *= 4./(9.*N*N);
    total = 1+(amplitude*total/sqrt(t));
    return_val = total;
    '''

support = \
    r'''
    double weight( int i, int j, int N )
        {
        if(i==1 || i==N+1)
            {
            if(j==1 || j==N+1) return(1.);
            if(j%2==0) return(4.);
            else return(2.);
            }
        if(i%2==0)
            {
            if(j==N+1) return(4.);
            if(j%2==0) return(16.);
            else return(8.);
            }
```

```
        else
            {
            if(j==N+1) return(2.);
            if(j%2==0) return(8.);
            else return(4.);
            }
        }

    double func( double z_1, double z_2, double v_x, double v_y, double v_z, double deltax,
double deltay, double deltaz, double t)
        {
        double z_r = 0.98; double w_0 = 0.5; double D = 220;
        double wsqr_1 = (w_0*w_0)*(1+(z_1/z_r)*(z_1/z_r));
        double wsqr_2 = (w_0*w_0)*(1+((z_2-deltaz)/z_r)*((z_2-deltaz)/z_r));
        return ((1/(wsqr_1*wsqr_2))*(1/(2*D*t + wsqr_1/8 + wsqr_2/8))*exp(-((v_x*t-
deltax)*(v_x*t-deltax))/(4*D*t + wsqr_1/4 + wsqr_2/4))*exp(-((v_y*t-deltay)*(v_y*t-
deltay))/(4*D*t + wsqr_1/4 + wsqr_2/4))*exp(-((z_2-z_1-v_z*t)*(z_2-z_1-v_z*t))/(4*D*t)));
        }
    '''

def int_simp2d_c(N,amp,velx,vely,velz,dx,dy,dz,t):
    total = weave.inline(code, ['N','amp', 'velx', 'vely', 'velz', 'dx','dy','dz','t',], headers=['<math.h>'],
support_code = support)
    return total

def plotsplan(i):       #Plotting function
            fig = plt.figure(i, (9,6))
            ax = fig.add_subplot(111)
            ax.set_xscale('log')
            plt.scatter(t, x[i*len(t):len(t)*(i+1)], color = 'red', marker = '^')
            plt.plot(t, qq[i*len(t):len(t)*(i+1)], color = 'blue', lw = 2)
            plt.grid()
            plt.show()

def VelDirectionChecker(po, x, t):
    veltestpx = po
    veltestnx = po
    veltestnx[16] = -veltestnx[16]
    if sum(errfunc(veltestnx, x, t)) < sum(errfunc(veltestpx, x, t)):
        po = veltestnx
    else:
        po = veltestpx
    veltestpy = po
    veltestny = po
    veltestny[17] = -veltestny[17]
```

```
    if sum(errfunc(veltestny, x, t)) < sum(errfunc(veltestpy, x, t)):
        po = veltestny
    else:
        po = veltestpy
    veltestpz = po
    veltestnz = po
    veltestnz[18] = -veltestnz[18]
    if sum(errfunc(veltestnz, x, t)) < sum(errfunc(veltestpz, x, t)):
        po = veltestnz
    else:
        po = veltestpz
    print po
    return po


start = time.time()
plsq = leastsq(errfunc, po, args = (x,t,w))[0]    #Fitting module
q = test_func(po, t)  #parameter for fitting
qq = test_func(plsq, t)
end = time.time() - start

print plsq

for i in range(16):
    plotsplan(i)
```

Appendix B. Python Code for Tracking

This python code reads in the spreadsheets output from the LabVIEW single emitter tracking program. The code reads in the positions, photon counts, and flags from the spreadsheet. The flags are then examined for when a tracking run begins and when it ends. The code then determines how the tracking run ended (lost, edge, or end cases) and the average time an emitter is tracked.

```
import numpy as np
import pylab as plt
from matplotlib import pyplot
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D

D = 0.5 #Diffusion constant in microns squared per second
diffusionouttime = 243
timeloop = 1.84 #Loop time for all four diodes to cycle
Resetcheck = 10 #Number of loops to wait begin new cycle trend
Threshold = 10 #Number of loops particle must be followed before analysis kicks in
Mintime = 10
TimesArray = np.array(0)
TimesArraymined = np.array(0)
LostParticle = 0
EdgeParticle = 0
EndParticle = 0
Longestrun = 0
Longestrunindex = 0
Longestrunfile = 0

for j in range(9):
    filenameyo = ('//OCTOPUS/Zepto/James Germann/2013/5-22-13 72% Glycerol-Water
Mix/Run ' + str(j+13) + ' 72% Mix.txt')
    f = open(filenameyo, 'r')
    data = []
    for line in f:
        data.append(line.split('\t'))
    f.close()

    data = np.array(data).astype(np.float)
```

```
    Ison = 0
    OnStart = 0
    print data[3][3]
    for i in range(len(data)):   #Turns on tracking
        if Ison == 0:
            if data[i][3] > 1:
                Ison = 1
                OnStart = i
                print 'On'
        if Ison == 1:           #If tracking is on
            if i == len(data)-1:   #Have you hit the end?
                if (i-OnStart) > Mintime:
                    EndParticle = EndParticle + 1
                    TimesArray = np.append(TimesArray, i - OnStart)
                    print 'Ended'
                Ison = 0
                if (i-OnStart) > Longestrun:
                    Longestrun = i - OnStart
                    Longestrunindex = i
                    Longestrunfile = j
            elif data[i][3] < 1:   #Have you hit the edge?
                if data[i-3][0] == 15.0 and data[i-3][1] == 15.0 and data[i-3][2] == 5.0:
                    if (i-OnStart) > Mintime:
                        print 'Edge'
                        EdgeParticle = EdgeParticle + 1
                        TimesArray = np.append(TimesArray, i - OnStart)
                    Ison = 0
                    if (i-OnStart) > Longestrun:
                        Longestrun = i - OnStart
                        Longestrunindex = i
                        Longestrunfile = j
                elif (len(data)) - i > Resetcheck:
                    total = 0
                    for k in range(Resetcheck):
                        total = total+data[i+k][3]
                    for j in range(Resetcheck):
                        if data[i+j][3] == 1 and (data[i+j][0]-data[i][0])**2+(data[i+j][1]-
data[i][1])**2+(data[i+j][2]-data[i][2])**2 > 4*D*np.sqrt(j+1)*timeloop*10**-3:   #Have you
lost the particle and picked up a new one
                            if (i-OnStart) > Mintime:
                                print 'Lost'
                                LostParticle = LostParticle + 1
                                TimesArray = np.append(TimesArray, i - OnStart)
                            Ison = 0
                            if (i-OnStart) > Longestrun:
```

```
                        Longestrun = i - OnStart
                        Longestrunindex = i
                        Longestrunfile = j
                    break
            if total < 1:   #Have you lost the particle and not picked up a new one
                if (i - OnStart) > Mintime:
                    print 'lost'
                    LostParticle = LostParticle + 1
                    TimesArray = np.append(TimesArray, i - OnStart)
                Ison = 0
                if (i-OnStart) > Longestrun:
                    Longestrun = i - OnStart
                    Longestrunindex = i
                    Longestrunfile = j
        elif (len(data) - i) < Resetcheck:
            total = 0
            for k in range(len(data)-i):
                total = total + data[i+k][3]
            for j in range(len(data) - i):
                if data[i+j][3] == 1 and (data[i+j][0]-data[i][0])**2+(data[i+j][1]-
data[i][1])**2+(data[i+j][2]-data[i][2])**2 > 4*D*np.sqrt(j+1)*timeloop*10**-3:   #Have you
lost the particle and picked up a new one
                    if (i - OnStart) > Mintime:
                        print 'Lost'
                        LostParticle = LostParticle + 1
                        TimesArray = np.append(TimesArray, i - OnStart)
                    Ison = 0
                    if (i-OnStart) > Longestrun:
                        Longestrun = i - OnStart
                        Longestrunindex = i
                        Longestrunfile = j
                    break
            if total < 1:   #Have you lost the particle and not picked up a new one
                if (i - OnStart) > Mintime:
                    print 'lost'
                    LostParticle = LostParticle + 1
                    TimesArray = np.append(TimesArray, i - OnStart)
                Ison = 0
                if (i-OnStart) > Longestrun:
                    Longestrun = i - OnStart
                    Longestrunindex = i
                    Longestrunfile = j
print LostParticle
print EdgeParticle
print EndParticle
```

```
np.delete(TimesArray, 0)
##print TimesArray
##print len(TimesArray)
print np.average(TimesArray)
##
Histobins = np.zeros(8)
for k in range(len(TimesArray)):
   if TimesArray[k]< diffusionouttime/timeloop:
      Histobins[0] = Histobins[0]+1
   elif TimesArray[k]>diffusionouttime/timeloop and
TimesArray[k]<2*diffusionouttime/timeloop:
      Histobins[1] = Histobins[1]+1
   elif TimesArray[k]>2*diffusionouttime/timeloop and
TimesArray[k]<5*diffusionouttime/timeloop:
      Histobins[2] = Histobins[2]+1
   elif TimesArray[k]>5*diffusionouttime/timeloop and
TimesArray[k]<10*diffusionouttime/timeloop:
      Histobins[3] = Histobins[3]+1
   elif TimesArray[k]>10*diffusionouttime/timeloop and
TimesArray[k]<20*diffusionouttime/timeloop:
      Histobins[4] = Histobins[4]+1
   elif TimesArray[k]>20*diffusionouttime/timeloop and
TimesArray[k]<50*diffusionouttime/timeloop:
      Histobins[5] = Histobins[5]+1
   elif TimesArray[k]>50*diffusionouttime/timeloop and
TimesArray[k]<100*diffusionouttime/timeloop:
      Histobins[6] = Histobins[6]+1
   else:
      Histobins[7] = Histobins[7]+1

Shortlist = []

for q in TimesArray:
   if q<15*diffusionouttime:
      Shortlist.append(q)


Histotimebins, bins = np.histogram(Shortlist, bins=25)
wtf=len(Histobins)
print Histotimebins
print bins
g = open('//OCTOPUS/Zepto/James Germann/2013/100Binsforgraphs2.txt','w')
for k in range(len(Histobins)):
   g.write('{ }\t{ }\n'.format(TimesArray[k], Histobins[k]))
for l in range(len(TimesArray)-len(Histobins)):
```

```
    g.write('{}\n'.format(TimesArray[wtf+l]))
g.close()

print Histobins
print Longestrun
print Longestrunindex
print Longestrunfile
```

VITA

James A. Germann was born in Portsmouth, Ohio, to parents Aloysius and Margaret Germann. He is the third of five children. In 2003 he graduated high school from Ralph L. Fike High School in Wilson, North Carolina. He then attended North Carolina State University in Raleigh, North Carolina, and graduated in 2007 with a Bachelors of Science degree in physics and a minor in mathematics. James continued his studies at the University of Tennessee Space Institute and received his Master of Science in physics the Fall of 2009.